Resolving Representation Heterogeneity in Real-World Knowledge Graphs

Von der
Carl-Friedrich-Gauß Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

von
Jan-Christoph Kalo
geboren am 12.05.1989
in Celle

Eingereicht am: 12.03.2021
Disputation am: 04.06.2021
1. Referent: Prof. Dr. Wolf-Tilo Balke
2. Referent: Prof. Dr. Felix Naumann

2021

Abstract

Knowledge graphs are repositories providing factual knowledge about entities. They are a great source of knowledge to support modern AI applications for Web search, question answering, digital assistants, and online shopping. The advantages of machine learning techniques and the Web's growth have led to colossal knowledge graphs with billions of facts about hundreds of millions of entities collected from a large variety of sources. While integrating independent knowledge sources promises rich information, it inherently leads to heterogeneities in representation due to a large variety of different conceptualizations. Thus, real-world knowledge graphs are threatened in their overall utility. Due to their sheer size, they are hardly manually curatable anymore. Automatic and semi-automatic methods are needed to cope with these vast knowledge repositories.

We first address the general topic of representation heterogeneity by surveying the problem throughout various data-intensive fields: databases, ontologies, and knowledge graphs. Different techniques for automatically resolving heterogeneity issues are presented and discussed, while several open problems are identified. Next, we focus on entity heterogeneity. We show that automatic matching techniques may run into quality problems when working in a multi-knowledge graph scenario due to incorrect transitive identity links. We present four techniques that can be used to improve the quality of arbitrary entity matching tools significantly. Concerning relation heterogeneity, we show that synonymous relations in knowledge graphs pose several difficulties in querying. Therefore, we resolve these heterogeneities with knowledge graph embeddings and by Horn rule mining. All methods detect synonymous relations in knowledge graphs with high quality. Furthermore, we present a novel technique for avoiding heterogeneity issues at query time using implicit knowledge storage. We show that large neural language models are a valuable source of knowledge that is queried similarly to knowledge graphs already solving several heterogeneity issues internally.

Zusammenfassung

Wissensgraphen sind eine wichtige Datenquelle von Entitätswissen. Sie unterstützen viele moderne KI-Anwendungen. Dazu gehören unter anderem Websuche, die automatische Beantwortung von Fragen, digitale Assistenten und Online-Shopping. Neue Errungenschaften im maschinellen Lernen und das außerordentliche Wachstum des Internets haben zu riesigen Wissensgraphen geführt. Diese umfassen häufig Milliarden von Fakten über Hunderte von Millionen von Entitäten; häufig aus vielen verschiedenen Quellen. Während die Integration unabhängiger Wissensquellen zu einer großen Informationsvielfalt führen kann, führt sie inhärent zu Heterogenitäten in der Wissensrepräsentation. Diese Heterogenität in den Daten gefährdet den praktischen Nutzen der Wissensgraphen. Durch ihre Größe lassen sich die Wissensgraphen allerdings nicht mehr manuell bereinigen. Dafür werden heutzutage häufig automatische und halbautomatische Methoden benötigt.

In dieser Arbeit befassen wir uns mit dem Thema Repräsentationsheterogenität. Wir klassifizieren Heterogenität entlang verschiedener Dimensionen und erläutern Heterogenitätsprobleme in Datenbanken, Ontologien und Wissensgraphen. Weiterhin geben wir einen knappen Überblick über verschiedene Techniken zur automatischen Lösung von Heterogenitätsproblemen. Im nächsten Kapitel beschäftigen wir uns mit Entitätsheterogenität. Wir zeigen Probleme auf, die in einem Multi-Wissensgraphen-Szenario aufgrund von fehlerhaften transitiven Links entstehen. Um diese Probleme zu lösen stellen wir vier Techniken vor, mit denen sich die Qualität beliebiger Entity-Alignment-Tools deutlich verbessern lässt. Wir zeigen, dass Relationsheterogenität in Wissensgraphen zu Problemen bei der Anfragenbeantwortung führen kann. Daher entwickeln wir verschiedene Methoden um synonyme Relationen zu finden. Eine der Methoden arbeitet mit hochdimensionalen Wissensgrapheinbettungen, die andere mit einem Rule Mining Ansatz. Beide Methoden können synonyme Relationen in Wissensgraphen mit hoher Qualität erkennen. Darüber hinaus stellen wir eine neuartige Technik zur Vermeidung von Heterogenitätsproblemen vor, bei der wir eine implizite Wissensrepräsentation verwenden. Wir zeigen, dass große neuronale Sprachmodelle eine wertvolle Wissensquelle sind, die ähnlich wie Wissensgraphen angefragt werden können. Im Sprachmodell selbst werden bereits viele der Heterogenitätsprobleme aufgelöst, so dass eine Anfrage heterogener Wissensgraphen möglich wird.

Publication List

This thesis is based on 4 publications* out of 11 publications I have published during my Ph.D. studies.

1. ***Jan-Christoph Kalo**, Silviu Homoceanu, Jewgeni Rose, and Wolf-Tilo Balke. Avoiding Chinese Whispers: Controlling End-to-end Join Quality in Linked Open Data Stores. ACM Web Science 2015 (WebSci), pages 1-10, 2015.
2. Stephan Mennicke, **Jan-Christoph Kalo**, and Wolf-Tilo Balke. Querying Graph Databases: What do Graph Patterns Mean? International Conference on Conceptual Modeling (ER), pages 134-148, 2017.
3. Stephan Mennicke, Denis Nagel, **Jan-Christoph Kalo**, Niklas Aumann, and Wolf-Tilo Balke. Reconstructing Graph Pattern Matches Using Sparql. Lernen Wissen Daten Analysen (LWDA), DB Workshop, pages 152-165, 2017.
4. **Jan-Christoph Kalo**, Christoph Lofi, René Pascal Maseli, and Wolf-Tilo Balke. Semantic Query Processing: Estimating Relational Purity. Lernen Wissen Daten Analysen (LWDA), DB Workshop, pages 113-125, 2017.
5. Stephan Mennicke, **Jan-Christoph Kalo**, and Wolf-Tilo Balke. Using Queries as Schema-templates For Graph Databases. Datenbank-Spektrum, pages 89-98, 2018.
6. Stephan Mennicke, **Jan-Christoph Kalo**, Denis Nagel, Hermann Kroll, and Wolf-Tilo Balke. Fast Dual Simulation Processing of Graph Database Queries. International Conference on Data Engineering (ICDE), pages 244-255, 2019.
7. ***Jan-Christoph Kalo**, Philipp Ehler, and Wolf-Tilo Balke. Knowledge Graph Consolidation by Unifying Synonymous Relationships. International Semantic Web Conference (ISWC), pages 276–292, 2019.
8. ***Jan-Christoph Kalo**, Stephan Mennicke, Philipp Ehler, and Wolf-Tilo Balke. Detecting Synonymous Properties by Shared Data-driven Definitions. Extended Semantic Web Conference (ESWC), pages 360–375, 2020.
9. ***Jan-Christoph Kalo**, Leandra Fichtel, Philipp Ehler, and Wolf-Tilo Balke. KnowlyBert - Hybrid Query Answering over Language Models and Knowledge Graphs. International Semantic Web Conference (ISWC), pages 294–310, 2020.
10. Hermann Kroll, **Jan-Christoph Kalo**, Denis Nagel, Stephan Mennicke, and Wolf-Tilo Balke. Context-compatible Information Fusion for Scientific Knowledge Graphs. International Conference on Theory and Practice of Digital Libraries (TPDL), pages 33–47, 2020.
11. Nitisha Jain, **Jan-Christoph Kalo**, Wolf-Tilo Balke, Ralf Krestel. Do Embeddings Actually Capture Knowledge Graph Semantics?. Extended Semantic Web Conference (ESWC), 2021, to appear

Acknowledgements

First and foremost, I would like to thank Wolf-Tilo Balke, who awoke my enthusiasm for information systems, guided me on my way to becoming a researcher, and gave me valuable feedback on this path. Right from the beginning, you were a great mentor who constantly challenged me. Also, I appreciate working with you on a personal level and always felt invited to your office. I would also like to thank Felix Naumann for taking the time to review my thesis.

I want to thank my colleagues from the Institute for Information Systems at the TU Braunschweig for the great working environment, the interesting discussions, and the motivating words. In particular, I would like to thank Stephan Mennicke for always being supportive and helping me to get a theoretical perspective on my ideas. I have learned a lot from you. I would like to thank Hermann Kroll for intense discussions, extremely helpful comments, and wonderful collaborations. I was happy to start the day with you two on WhatsApp even before we met in the office the same day. I will really miss that. Furthermore, I would like to thank my research assistants and co-authors Philipp Ehler and Leandra Fichtel, for supporting me 24/7 when it was needed. Without you two, finishing some papers on time would have been really difficult.

And of course, I am also thankful for the helpful and critical discussions with my colleagues and students Janus Wawrzinek, José Pinto, Kinda El Maarry, Silviu Homoceanu, Christoph Lofi, Denis Nagel, and René Maseli for making working at IfIS a great experience. Thanks for the many great coffee breaks we spend together.

Furthermore, I would like to thank Regine Dalkiran for many interesting conversations, for great movie recommendations, and for always helping me out.

Finally, I am grateful to Anna for always supporting me during difficult times. I would like to thank my parents for making it possible to visit a university and always being there for me.

Table of Contents

1 Introduction

In 1999, the World Wide Web (WWW) inventor, Tim Berners-Lee, has first sketched ideas of a Semantic Web in contrast to the classical WWW as we know it [7]. This idea was further explained in the seminal article in the Scientific American "The Semantic Web". The article describes a semantically connected Web, where agents can automatically understand websites to provide intelligent services to end users [8]. As an example, Berners-Lee chose an intelligent agent to automatically make a doctor's appointment. It uses and integrates services from several websites to choose the right doctor within the area, having excellent reviews on the Web. Furthermore, it should check the patient's calendar to find her schedule when making the appointment. In the classical WWW, offering such a service is usually impossible since knowledge on the Web is expressed in a human-readable format. Websites usually contain texts, tables, photos, diagrams, and videos. None of it is fully understood by computers.

To overcome this problem, representing knowledge in a machine-readable format on the Web was introduced. The Resource Description Framework (RDF) allows for representing knowledge on the Semantic Web: Each piece of knowledge is a fact that is stored as a *subject, predicate, object* triple [24]. For example, (`Albert Einstein, bornIn, Ulm`) is a triple that expresses knowledge about Einstein's place of birth Ulm. Each triple can be depicted as a graph with two nodes connected by an edge. Hence, several triples together form a large graph, a so-called *knowledge graph*.

From early on, database research has addressed the possibility of modeling semantically equivalent information from real-world in different ways. Since natural language descriptions allow for a large variety of expressing knowledge, and the actual selection of some data model and schema also strongly depend on the focus and requirements of the respective application, *representation heterogeneity* is bound to emerge in the forms of *syntactic heterogeneity*, *schematic heterogeneity*, and *semantic heterogeneity* [75, 86]. Thus, when merging data instances from multiple sources, several heterogeneity problems may come up. These problems may include synonymous/homonymous use of class names and attributes. Schemas may differ on their level of detail. Furthermore, a variety of other structural/semantic conflicts may arise. To overcome such issues, *schema matching and entity matching methods* allow

to smoothly integrate data from different sources either manually or with increasing degrees of automation are essential.

While classical matching and alignment problems mostly operated on clear-cut database schemas (usually relational tables with fixed sets of attributes and data types [96]), in the Semantic Web, knowledge is not stored in relations, but in a much more unstructured way using XML or RDF. Imprinting semantic structure on such data usually means creating complex ontologies using the Web Ontology Language (OWL), which in turn drove the need for new techniques for ontology matching. During the early years of the Semantic Web, the vision of an integrated Linked Open Data (LOD[1]) cloud with thousands of heterogeneous and unstructured data sources was born. This vast amount of heterogeneous data sources introduced serious concerns for algorithmic scalability. The yearly benchmarks at the Workshop for Ontology Alignment at the International Semantic Web Conferences (Ontology Alignment Evaluation Initiative[2]) clearly show the importance of respective research. Today, the goal of largely integrated semi-structured data sets is still not reached. While the vision of a LOD cloud has recently lost attention, knowledge graphs have gained a lot of attention because of the popularity of projects, such as the Google Knowledge Graph [27], Wikidata [113], DBpedia [4], Freebase [11], and YAGO [103]. The size of these knowledge graphs nowadays comprises hundreds of millions of entities associated by ten thousands of properties, providing a comprehensive knowledge repository for several modern applications, e. g., semantic search, question answering, product search, social networks, and even natural language understanding [85].

The size of these knowledge graphs has steadily grown over the last years. Often large knowledge graphs are created manually in collaborative knowledge graph projects [113], automatically by extracting information from text or tables [4, 27], by integrating existing knowledge into a single knowledge graph, or by any combination of all these methods. However, integrating knowledge from various sources and by different curators into a single knowledge graph comes with serious heterogeneity issues in practice. Particularly, duplicate concepts, either entities, classes, or properties, may cause problems in subsequent querying. In fact, the knowledge graphs' semantic heterogeneity combines most of the previous applications' problems:

- *Algorithmic scalability:* knowledge graphs are huge in size, comprising hundreds of millions of entities and billions of facts.
- *Degree of heterogeneity:* knowledge graphs are built from a huge number of diverse data sources. Often they even range across multiple languages and domains, adding an additional layer of complexity.
- *Structural issues and level of detail:* heterogeneity issues often combine entity, relation, class, and literal heterogeneity at once.

[1] `https://lod-cloud.net/`
[2] `http://oaei.ontologymatching.org/`

Contributions. This thesis investigates the heterogeneity issues in large-scale heterogeneous real-world knowledge graphs, such as Wikidata, DBpedia, and Freebase. We give an extensive overview of the problem of heterogeneity issues in different data representation formats, such as relational databases, ontologies, and knowledge graphs. While we discuss that heterogeneity is a long-standing problem, we also introduce the challenges that come with large real-world knowledge graphs. We survey state-of-the-art methods for resolving heterogeneity problems, such as *entity matching, relation matching, class matching, literal matching, and ontology matching*. Finally, we point out unsolved heterogeneity issues that are further investigated throughout the thesis.

We perform an extensive study of *entity heterogeneity in multi-knowledge graph matching problems*. Our work explicitly analyzes problems of `owl:SameAs` identity links in entity matching scenarios with multiple knowledge graphs on a large scale. Our work includes four methods for supporting arbitrary entity matching systems when working in a multi-knowledge graph scenario to improve their matching quality. To evaluate our new techniques, we introduce a novel benchmark dataset for instance matching systems consisting of seven different knowledge graphs, and made it publicly available. In the experiments, we show that, indeed, our proposed methods improve instance matching by more than 10% in precision without losing any recall.

This contribution was published at the *ACM Web Science Conference 2015* [57]. The basic ideas of this work were developed during my Master thesis [54].

We show that *synonymous relations* are an issue in knowledge graphs that got hardly any attention, even though it may cause incomplete query results. To overcome these issues, we present several techniques that identify synonymous relations in large-scale knowledge graphs. Both our techniques are purely data-driven and make no requirements on the data. One technique is based on state-of-the-art knowledge graph embeddings and outlier detection. We are able to detect synonymous relations with very high precision.

To overcome explainability issues in our first approach, we also present a technique using Horn rule mining to find logical definitions of relations. By an indirect rule mining approach, we are able to identify synonymous relations with higher precision than the embedding-based techniques and provide human-understandable explanations.

Our techniques are evaluated on synthetic benchmark datasets for synonym detection and a large manual evaluation on Freebase, Wikidata, and DBpedia. The datasets and the code of our methods are openly available to guarantee the reproducibility of the results.

These contributions were published in two papers: The first one was published at the *International Semantic Web Conference 2019* [55], the second one about using Horn rule mining at the *Extended Semantic Web Conference 2020* [58].

Our final contribution is a first analysis of supporting heterogeneous and incomplete knowledge graphs by neural language models. We show that combining knowledge graphs with a language model as an implicit knowledge store leads to valuable

improvements when querying heterogeneous or incomplete knowledge graphs since a language model is able to overcome heterogeneity issues that are present in natural language text.

We show that heterogeneous knowledge graphs profit from language models to return complete result sets while querying language models profit from the valuable semantic information in knowledge graphs. In our experiments on more than 6.000 queries, we have shown that, indeed, the combination of implicit and explicit knowledge stores is a valuable resource for query answering. Queries could be completed with a precision of almost 50% precision. The implementation and data of this work are available for reproducibility purposes.

Our system *KnowlyBert* was published at the *International Semantic Web Conference 2020* [56].

Outline. This work is structured into four larger chapters describing various aspects of heterogeneity issues. Furthermore, we propose several novel approaches on how to deal with them.

First, we give an extensive introduction to the topic of representation heterogeneity in knowledge graphs in Chapter 2. It starts with an introduction to Semantic Web technologies and a definition of knowledge graphs in Section 2.1 Next, we introduce semantic heterogeneity in general and show where different aspects of heterogeneity have been tackled in Section 2.2, 2.3, and 2.4. We explain the novel challenges that come with knowledge graphs and show how current techniques solve these issues in Section 2.5 and 2.6. At the end of the chapter, we summarize the current state of heterogeneity issues in knowledge graphs and show open problems that we further investigate throughout this thesis.

One of these problems, entity heterogeneity in multi-knowledge graph scenarios, is further discussed in Chapter 3. We start by discussing the related work of entity heterogeneity for multiple knowledge graphs and stress the differences to standard entity matching with exactly two data sources in Section 3.1. Next, we introduce the *Chinese Whispers problem* in entity heterogeneity and propose four different approaches for improving standard instance matching systems in multi-knowledge graph environments in Sections 3.2 and 3.3. Finally, we evaluate our methods on two state-of-the-art instance matching systems on a large benchmark, which we manually curated for this problem in Section 3.4.

In Chapter 4, we further discuss the problem of relation heterogeneity within single knowledge graphs. Concretely, our focus is on *synonymous relations*. We show how synonymous relations are different from standard heterogeneity issues that are tackled by matching systems in Section 4.1. We then propose two different approaches in Section 4.2 and Section 4.3: (1) A knowledge graph embedding-based approach which employs eight state-of-the-art embedding techniques to identify synonymous relations in large knowledge graphs, and (2) and a synonym detection technique using Horn rule mining. The rule-based approach offers great explainability while guaranteeing high precision results. In the conclusion, we discuss the current state of synonymous relation detection in large-scale knowledge graphs.

Chapter 5 explains how heterogeneity issues in knowledge graphs are avoided by storing knowledge implicitly in novel neural language models. The related work in

this direction is shortly discussed in Section 5.1. We first, in Section 5.2, introduce the idea of how language models are used as an implicit knowledge graph and how queries are performed. To test our hypothesis that heterogeneity is partly solved by implicit knowledge storage in language models, we build a hybrid querying system, KnowlyBERT, in Section 5.3. It combines language models with knowledge graphs, such that both techniques profit from each other. KnowlyBERT is evaluated on thousands of entity-centric queries and shows promising results in Section 5.4

Finally, in Chapter 6, we discuss the outcomes of this thesis. We first conclude each chapter separately from each other and, in the end, conclude on the current state of heterogeneity issues in knowledge graphs. Furthermore, we develop ideas for future work.

2

Representation Heterogeneity in Knowledge Graphs

Natural language is an essential component for humans to represent knowledge. Humans use language for representing knowledge in texts or to communicate with each other in written or spoken form. Furthermore, many researchers believe that language strongly influences what we think and how we think. In language, a piece of knowledge, i.e., a fact, can be represented in various heterogeneous ways. If asking two persons to express the fact that Albert Einstein was born in Ulm in one sentence, we may end up with totally different sentences: *"Albert Einstein is born in Ulm."* and *"The birthplace of the German scientist Einstein is Ulm."*. Both carry similar meanings but are expressing the information differently. Both refer to the German scientist *Albert Einstein*, but only the last name is used in one sentence. Hence, the representation of this entity is different. A more striking difference is the expression of the `birthplace` relation. In the first sentence, the verb *born* is used, while the second sentence uses the noun *birthplace*. Besides this small example, natural language offers a variety of more complex heterogeneities. For example, complex sentence constructions may be used to encode the same piece of knowledge.

Heterogeneity in natural language knowledge representation carries over to structured knowledge representations, e.g., formal knowledge representation techniques, such as RDF, or simply relational databases. Various data-intensive fields of computer science have noticed such heterogeneity problems when working with knowledge. When we ask persons to create a structured data model for storing information about persons, the results probably look different. These differences involve linguistic differences but also structural differences.

Example. These two models about persons may look as follows: Each model may consist of precisely one table: *Persons* and *People*, each with a set of attributes describing Albert Einstein. The *Person* table contains a single entry about Einstein, his address, and birth date. *People* comprises two entries, each about Albert Einstein, but with information regarding his name, birthday and birthplace. Both tables contain partly identical information: we immediately spot several data interoperability

problems here, such as a naming conflict (i.e., synonyms) on table level (person vs. people), similarly on attribute level (birthdate vs. birthday). There are structural differences on attribute level regarding the granularity of information (firstname and lastname vs. name), different date formats for the birth date, different attribute granularities for birthplace and birthdate, and a different name format. Additionally, the *People* table contains a duplicate entry for the same real-world object. For larger data models, we may have several additional structural heterogeneity issues, e.g., different normalization levels.

Table 2.1: Two example tables with different attributes, but both representing information about persons.

Person	**Firstname**	**Lastname**	**Address**	**Birthdate**
	Albert	Einstein	Adlzreiterstraße 12, München	March 14, 1879

People	**Name**	**Birthday**	**Birthplace**
	Albert Einstein	14.03.1879	Ulm
	Einstein, Albert	1879	Germany

The sheer variety of different design options for single facts has been outlined in [60]. This work shows that a simple binary relation between a salesman and his territory may be modeled in at least 36 different ways using a simple relational model. All these models are straight-forward models of the same fact. They may come up in practical scenarios, causing problems when interchanging data. Similar to relational databases, these kinds of heterogeneity issues may also evolve in every other representation of data and knowledge.

Research Questions.

- *How do we classify heterogeneity problems?*
- *How is heterogeneity in knowledge graphs different to classical databases?*
- *How are existing techniques to tackle heterogeneity in knowledge graphs different to each other?*
- *What are open heterogeneity issues in knowledge graphs?*

Contribution. Throughout this chapter, we provide an introduction to data heterogeneity in general and provide an idea of what knowledge graphs are. We survey the problem of heterogeneity in various fields, from databases to ontologies and knowledge graphs. With this, we stress problems that come with large-scale heterogeneous knowledge graphs. We characterize existing solutions for heterogeneities in knowledge graphs into four categories: entity heterogeneity, relation heterogeneity, class heterogeneity, and literal heterogeneity. Within these categories, we compare a wide variety of techniques and discuss their advantages and disadvantages. Furthermore, we describe open problems in current research on heterogeneous knowledge graphs.

Outline. Section 2.1 provides an introduction to the Semantic Web idea and the various technologies (RDF, RDF-S, OWL, and SPARQL). Furthermore, we give a definition of knowledge graphs. Section 2.2 gives an overview of representation heterogeneity and provides different classification systems for it. A short introduction to heterogeneity issues in relational databases is given in Section 2.3 Section 2.4 describes heterogeneity issues in ontologies. In Section 2.5, we give an overview of heterogeneity issues in knowledge graphs and how they are different from heterogeneity issues in databases and ontologies. Then, we go into the details of entity, relation, literal, and class heterogeneity issues. Several matching approaches for the different heterogeneity issues in knowledge graphs are surveyed in Section 2.6. Finally, in Section 2.7, we shortly conclude state-of-the-art matching systems and describe some open problems.

2.1 RDF, SPARQL, and Knowledge Graphs

To overcome the shortcomings of existing Web technologies for achieving a Semantic Web, Berners-Lee and other Semantic Web researchers have introduced various new technologies to annotate Web sites with semantic knowledge in a machine-readable format using knowledge representation technologies. The Semantic Web idea involves various technologies that can be presented as the Semantic Web stack (cf. Figure 2.1). The stack involves technologies for encoding and representing knowledge, querying and reasoning, and some layers that have not been implemented (cryptography and trust features). In this work, we mainly focus on the essential parts for knowledge representation and reasoning: RDF, RDF-S, OWL, and SPARQL.

RDF. The core technology, which also became a W3C standard for expressing knowledge on the Web, is the *Resource Description Framework*(RDF) [24]. At its center is the idea that knowledge is expressed in `subject, predicate, object` facts. The facts are often also called *triples*.

Example. As an example, a triple expressing the fact that Albert Einstein was born in the city Ulm may be expressed as follows:

```
(Albert Einstein, bornIn, Ulm)
```

In this example, the scientist `Albert Einstein` is a subject. The verb from the natural language sentence is a predicate `bornIn`, and the city of `Ulm` is an object in the triple.

Due to the triple format, facts may also be represented as a graph where subjects and objects are nodes. The predicate may be represented as an edge connecting the respective nodes as presented in Figure 2.2.

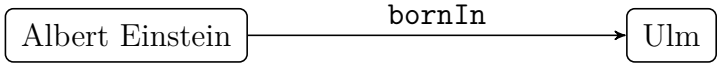

Figure 2.2: A simple graph representation of the triple (Albert Einstein, bornIn, Ulm).

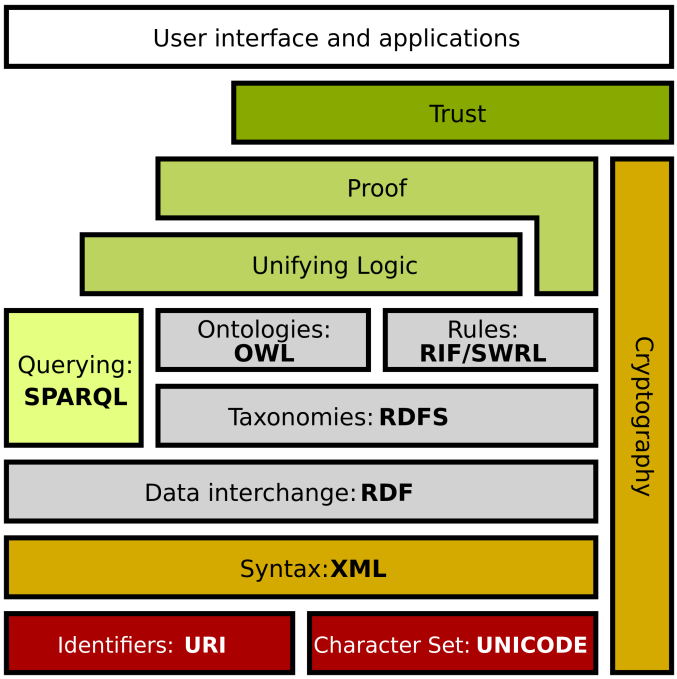

Figure 2.1: An illustration of the Semantic Web stack describing different technologies involved in the Semantic Web from [107]. On the lower levels, we have representation technologies, on the middle levels technologies for reasoning, and on the higher levels ideas that have not been yet realized.

Since RDF is a Web technology that should connect knowledge between various Web sites, subject, predicates, and objects are unique identifiers, *Internationalized Resource Identifiers* (IRI), instead of natural language names. As an example, for Albert Einstein, an IRI which is found in the RDF dataset Wikidata is `www.wikidata.org/entity/Q937`, the IRI for the `bornIn` relation is `www.wikidata.org/prop/P19`. An IRI may consist of a prefix `www.wikidata.org/entity/` and a unique identifier for the dataset `Q937`. Since IRIs usually impair readability, in this work, we solely work with natural language labels instead.

More formally, an RDF triple may be defined as follows: $(s, p, o) \in E \times R \times (E \cup L)$. Subjects are from a set of resources E. They usually represent entities or concepts (often from the real-world). Predicates stem from a set of relations R. The object either is a resource similar to the subject or a literal from the set L. In contrast to resources and relations, literals are not represented by IRIs but may be strings, numbers, or dates. As an example, a triple concerning a literal may be about Einstein's birthday:

```
(Albert Einstein, birthDate, '14 March 1879')
```

In logics, a relation is seen as a binary predicate: $\texttt{bornIn}(AlbertEinstein, Ulm)$. In this work, we also use the mathematical term (binary) *relation* over subjects and objects to describe triples. Note that we focus on RDF without blank nodes and reification.

RDF-Schema. Schematic information in RDF may be expressed by the *Resource Description Framework - Schema* (RDF-S) [15]. RDF-S provides a fixed vocabulary for expressing schematic information about RDF data by annotating resources and relations properly. Hence it is possible to describe groups of resources or unique resources. The maybe most used RDF-S features is the `rdfs:label` for giving natural language labels to resources or relations.

```
(www.wikidata.org/entity/Q937, rdfs:label, 'Albert Einstein')
```

Another critical feature of RDF-S is used to construct groups of resources, so-called *classes* and the idea to build *class hierarchies* for expressing complex knowledge taxonomies. Here, the RDF-S vocabulary `rdfs:type` and `rdfs:SubClassOf` are used:

```
(Albert Einstein, rdfs:type, Scientist)
(Scientist, rdfs:SubClassOf, Person)
(Scientist, rdfs:type, rdfs:Class)
(Person, rdfs:type, rdfs:Class)
```

The two triples express that Einstein is of the type of `scientist`. A `scientist` is describing a group of entities, also called *class*. Furthermore, we have expressed that the class `scientist` is a subclass of the class `person` building a class hierarchy. Class hierarchies are frequently used in Semantic Web datasets to formalize conceptual knowledge.

OWL. The formal representation of knowledge is often done in an *ontology*. An ontology is a formal way to describe knowledge as concepts, categories, properties, and relations. To go from basic schema information to the more complex idea of ontologies, the *Web Ontology Language (OWL)* was introduced [70]. OWL provides a vocabulary for annotating RDF with schema information. However, much more complex semantic expressions are possible. As an example, cardinalities and restrictions for relations or literals may be expressed. Also, complex relations between classes, such as intersections, unions, and complements, are possible. Furthermore, OWL offers a wide range of possibilities for reasoning. OWL reasoning is used to infer new knowledge based on existing knowledge using logical entailment rules. However, in this work, we do not go into further details of OWL reasoning capabilities.

An essential feature of OWL for this work is expressing identity between resources, relations, and classes. Here, the OWL vocabulary offers the three relations `owl:sameAs`, `owl:equivalentProperty` and `owl:equivalentClass`. The property `owl:equivalentProperty` is used to express that two relations have the same extension. That means the relations are used for precisely the same resources. Similarly the property `owl:equivalentClass` is used for classes.

However, both relations are not used to express the equality of classes and relations. The property for expressing identity between individuals is `owl:sameAs`. It may express the equality between resources, relations, and classes when they have the same real-world semantics.

SPARQL. The querying language for RDF data is called *SPARQL* [92]. In this work, we restrict ourselves to the basic querying mechanism of SPARQL, basic graph patterns (BGP). Like the relational database query language SQL, each query in SPARQL consists of a `SELECT` and WHERE clause. The `SELECT` clause defines the projection variables. The selection criteria for the query are defined in the `WHERE`-clause as a set of triple patterns including variables, the BGP.

Example. As an example, we first show a short SPARQL query, asking for the birthplaces of all scientists from Germany.

```
SELECT ?birthplace
WHERE (
?person <bornIn> ?birthplace.
?birthplace <country> <Germany>.
?person <occupation> <Scientist>.
)
```

The BGP in the WHERE-clause consists of three triples containing variables indicated by a leading question mark and entities/relations indicated by angle brackets. The first triple asks for a ?*person* born in some ?*birthplace*. This ?*birthplace* should be in the `country` `Germany`. The ?*person* should have the `occupation` `Scientist`. Note that the naming of these variables is not carrying any semantics. The BGP triples are matched to the knowledge graph, matching each triple to a triple in the knowledge graph, such that variables with the same name are mapped to the same entities.

2.1.1 Knowledge Graphs

The term *knowledge graph* has been shaped by the idea of the Google Knowledge Graph, first mentioned in a blog article by Google in 2012 [100]. However, a clear definition is still missing. The term knowledge graph often is used interchangeably with the term ontology since they both often work with RDF and use both, classes and class hierarchies. Erlinger and Wöß have reviewed several definitions of knowledge graphs to come up with a unifying definition [28]:

> A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.

Generally speaking, they say that knowledge graphs are similar to classical ontologies. Particularly the idea of Linked Open Data is closely connected to the presented definition of knowledge graphs. Hence, several solutions from the field of ontology matching can be directly carried over to knowledge graphs.

An essential property of knowledge graphs that is not necessarily inherent in ontologies is that they have often been classified as being *very large* and usually integrate knowledge *from various data sources*. Thus, the techniques for solving heterogeneities need to be extremely scalable, being able to deal with hundreds of millions of entities. A *knowledge graph* is a finite set of triples $KG \subseteq E \times R \times (E \cup L)$.

2.2 Different Types of Heterogeneity

Representation heterogeneity itself is a long-standing problem in several fields of computer science. Throughout the previous decades, many definitions and classifications for different heterogeneity types have been proposed and discussed in the literature. Many of the existing classification schemes are more or less distinguishing between *syntactic heterogeneity*, *structural heterogeneity*, and *semantic heterogeneity* [75, 86].

Syntactic Heterogeneity usually occurs when we have different data sources using different representation formalisms or technologies. It might occur when one data source, for example, is represented in a relational database. The other one is represented in RDF. Mapping rules can often overcome syntactic heterogeneity from one formalism into the other.

Structural Heterogeneity comprises schematic differences between various data sources. Even for the same data model, e.g., the same entity-relationship model, we may have different structural schemas. It may involve different table structures, knowledge being represented as a relation or an attribute, different names, and more.

Semantic Heterogeneity is usually even more complex to detect than the previous types of heterogeneity because it involves the meaning of the different attributes and relations. Typical examples for semantic heterogeneities are synonyms and homonyms. Attributes have different labels but the same meaning or attributes from two schemas having the same label but with a different meaning.

Usually, these problems do not occur separately from each other but together. We further concentrate on structural and semantic heterogeneity issues, both usually going hand in hand.

To overcome heterogeneity issues in practice, usually, so-called *data integration* is performed. Data integration is the process of merging multiple heterogeneous data sources into a single schema while resolving existing heterogeneity issues. In data integration, usually two orthogonal problem classification types are used: *schema heterogeneity* and *data heterogeneity* [40, 61]. Schema heterogeneities in relational databases comprise conflicts in tables and attributes. Data heterogeneities involve differences in concrete data instances of a database. Note that schema, as well as data heterogeneity, can both be of syntactic, structural, and semantic nature.

Example. Looking at the two tables about Albert Einstein (cf. Table 2.2), schema heterogeneity involves structural and semantic heterogeneities. On the structural side, both tables have different attribute combinations, but both are about persons. The name in the people table is only a single attribute, while it is split into two attributes in the person table. With regard to semantic heterogeneity, we have several issues as well. Both tables have synonymous names (people and person) and synonymous attributes, as already discussed before.

Data heterogeneity, on the other hand, is about concrete data instances. All three entries describe the same real-world entity, Albert Einstein, whereas the people table

Table 2.2: Repetition from Table 2.1 for better readability. Two example tables with different attributes, but both representing information about persons.

Person	**Firstname**	**Lastname**	**Address**	**Birthdate**
	Albert	Einstein	Adlzreiterstraße 12, München	March 14, 1879

People	**Name**	**Birthday**	**Birthplace**
	Albert Einstein	14.03.1879	Ulm
	Einstein, Albert	1879	Germany

even contains two Einstein instances. On the structural side, instances from different tables, of course, have a heterogeneous structure. However, also the two duplicate instances from the people table have different data formats for name, birthday, and birthplace. Thus, also within the table itself, we have structural heterogeneities.

For resolving schema and data heterogeneity, two matching problems have usually been defined, which both are known under various terms. Note that these terms might be interpreted slightly differently by various authors in the literature.

Schema Matching is about resolving structural heterogeneities in two different data models by matching corresponding relations/tables and attributes [96]. For our example, a schema matching would identify that both tables, people, and persons describe the same real-world entity type and identify which column corresponds to which column in the other table.

Entity Matching on the other hand, is a technique for resolving data heterogeneity issues. Among many others, entity matching is also known under the terms *entity alignment*, *entity resolution*, *duplicate detection*, *record linkage*, or *instance matching*. Concretely, entity matching is about identifying data instances that refer to the same real-world entity. Given our example tables, entity matching would be about identifying entries in the two tables representing the same real-world entity, e.g., three Albert Einstein instances.

We also distinguish matching problems between *inter-source duplicates* and *intra-source duplicates* [76]. Intra-source duplicates are duplicates within the same source. It usually implies that the data entries already have a similar schema, which significantly reduces the number of schematic heterogeneities. However, heterogeneity problems, for example, involve typos, abbreviations, and different names for the same object. For example, we again look at the two tables about Einstein. The people table contains intra-source duplicates, so two entries about the same real-world object.

On the other hand, inter-source duplicates are about data heterogeneity between two separate data sources, usually having different schemas. Thus, the variety of heterogeneity issues is much larger. It may involve schematic and semantic heterogeneity issues so that a schema matching step needs to be performed before the actual entity matching is done. These inter-source duplicates are found among

our example people and person table since both contain an entry about the entity Albert Einstein.

Throughout this thesis, we look at both schema matching and entity matching in various settings. We now start by discussing the problems in relational databases and ontologies and then show how both problems occur in real-world knowledge graphs.

2.3 Heterogeneity Issues in Databases

Semantic heterogeneity in database schemas has become an essential field of research, especially with the introduction of large data warehouses trying to integrate all companies' data to perform extensive statistical data analysis for decision support. Good surveys are the following: [9, 96].

Schema matching usually was (and still often is today) a manual task performed by domain experts, often supported by graphical user interfaces. Early techniques to support this labor-intensive task were based on pre-defined sets of matching rules that could be extended manually by additional rules so that an automatic matching process was possible [73]. More recent techniques rely either on a mix of label similarities and structural similarity metrics [96] or use machine learning-based approaches that need some pre-aligned examples [6].

Following [96], matching approaches are classified into several classes. Schema matching is either performed in an instance-driven way by first matching the individual tuples of two schemas. So Einstein and Albert Einstein from the two tables above would first be matched. Building on top of the matched tuples, we perform schema matching and detect that the two tables should be aligned. The name attribute is matched to the respective first- and last name columns. In contrast, pure schema-based matching is differentiated by the usage of element-level or structural similarity metrics. Element-level similarity directly matches attributes and tables using name similarity and type similarity measures. Here, the similarity between birthday and birth date is used to find the correspondence. On the other hand, structural similarity measures are often more complex and work on the database schema's overall relation structure.

Few state-of-the-art works on schema matching are available. Some techniques mix human experts with automatic matching techniques so that the human effort is minimized. Still, high-quality alignments are found [80]. Another current trend in schema matching is focused on integrating data lakes, so multiple big datasets [3]. Techniques for this holistic schema matching have a significant focus on scalability and usually work with supervised learning techniques.

As discussed, entity matching also tries to find meaningful correspondences between the instance data. However, usually, entity matching and schema matching are data integration problems that occur simultaneously. [39]

Similar to schema matching, the process for entity matching first heavily relied on manually defined matching rules. This process was later automatized into an automatic rule mining approach. More recent approaches usually use various string

metrics to identify similar entries in multiple databases [114]. Today hundreds of different techniques for solving entity matching exist. Most techniques, however, still require domain experts for some kind of manual input [62]. Therefore, the current trend is building machine learning-based systems for matching that require small amounts of training data to automatically learn what a good matching is [62, 74].

2.4 Heterogeneity in Ontologies

Large ontologies in RDF/RDF-S/OWL have become more common during the early days of the Semantic Web. In ontologies, we are similarly confronted with schematic and semantic heterogeneity. In contrast to relational databases, the knowledge representation framework RDF offers a much larger degree of freedom when representing knowledge. Hence structural heterogeneity issues are much more prevalent. While classical databases use relations (or tables) with fixed sets of attributes for each entity type, RDF entities usually do not have a fixed schema. They may be used in all kinds of triples using arbitrary predicates without being restricted by such constraints. While this offers much more flexibility, which is needed in the Semantic Web context, it comes with additional heterogeneity issues.

Example. Similar to our previous example, we demonstrate how information about Einstein similar to Table 2.2 may be represented in RDF. However, we introduced missing information for the second table by removing the second Albert Einstein entry's birthdate. Each entity in RDF needs a unique identifier, an IRI. For this example, we used simple example IRIs, which are not conform with standard IRIs for readability reasons.

```
(Albert_Einstein_Person, Firstname, 'Albert')
(Albert_Einstein_Person, Lastname, 'Einstein')
(Albert_Einstein_Person, Address, 'Adlzreiterstraße 12, München')
(Albert_Einstein_Person, Birthdate, 'March 14, 1879')
(Albert_Einstein_People_1, name, 'Albert Einstein')
(Albert_Einstein_People_1, Birthday, '14.03.1879')
(Albert_Einstein_People_1, Birthplace,  Ulm_People_1)
(Albert_Einstein_People_2, Name,  'Einstein, Albert')
(Albert_Einstein_People_2, Birthplace,  Germany_People_1)
```

Similar to before, we have several structural and semantic heterogeneity issues involving synonyms, different attribute formats (e.g., March 14, 1879, vs. 14.03.1879). The most striking difference from ontologies to relational databases is the lack of fixed table structures for entities of the same type. Even entities of the same type (from the former People table) have different attributes, i.e., they have no common structure anymore. This unstructured nature of RDF makes matching problems significantly more difficult.

As mentioned above, OWL offers a concrete vocabulary to express the correspondence of different data items with related or identical meanings. `owl:sameAs` and `owl:equivalentClass` are predicates to express identity arbitrary data instances and

classes. However, OWL also offers the expression of more complex correspondences than just equivalence: consequence, subsumption, and disjointness [30]. These additional correspondences add a layer of complexity which rarely is covered in traditional schema matching.

2.4.1 Ontology Matching

Data integration of two or more ontologies is called *ontology matching*. Ontology matching usually refers to aligning all objects of an ontology: entities, relations, and classes. [30] Hence, similar to before, we are confronted with schema matching and entity matching problems in ontologies. Instead of entity matching, in ontologies, we often talk about *instance matching*.

To measure the quality of ontology matching systems, in 2004, a series of ontology matching benchmarks was first published as a contest[1]. Since 2006 the benchmark is hosted by the Ontology Matching Workshop co-located with the International Semantic Web Conference. In 2020, this benchmark comprised 13 different tracks with synthetic and real-world matching problems from several domains and with a different focus. In some of these matching tasks, systems are evaluated on their performance for simple and complex correspondences.

Since many ontology matching approaches are also used for data integration in knowledge graphs, we do not go into the details of approaches here but in Section 2.6. For a detailed overview of ontology matching techniques, the Ontology Matching book by Euzenat and Shvaiko gives an excellent overview [30].

2.4.2 Identity in Ontologies

One of the most common issues in ontologies is instance matching. The Web Ontology Language offers the `owl:SameAs` predicate for expressing that two IRIs are identical. The predicate may be used to express identity among entities, relations, or classes. However, expressing identity between entities is most common.

The usage of `owl:SameAs` has also led to lots of discussions because it often is misused [41]. Halpin et al. make a great effort to discuss identity on the Semantic Web, presenting typical issues when misusing `owl:SameAs` and analyzing its usage in Semantic Web data sources. They conclude that the usage is unclear to many users, and more precise guidance and documentation could improve the quality of identity links on the Semantic Web. For example, users often use `owl:SameAs` for very similar instances or instances that are not identical in all contexts.

Since identity management is a complex problem in a distributed environment, such as the Semantic Web, the idea of central entity naming systems came up [13]. The idea of OKKAM was to provide a unique identifier for each entity available across all semantic web data sources. However, OKKAM, as a central naming system, never got popular and did not solve the identity problem in the Semantic Web. The service was discontinued, and the identity links are not available anymore.

[1] `http://oaei.ontologymatching.org/`

Until today, discussions on `owl:SameAs` and entity matchings in ontologies are prevalent. However, today often knowledge graphs instead of ontologies are considered. For a detailed discussion on `owl:SameAs` refer to the survey by Raad et al. [95].

2.5 Heterogeneity in Knowledge Graphs

The previous section has provided a broad introduction to the available related work of semantic and structural heterogeneity in databases and ontologies. As already discussed before, knowledge graphs are similar to ontologies. Thus, heterogeneity issues are very similar to what we have already seen for ontologies.

However, knowledge graphs are seen as significantly larger ontologies, often from multiple sources and domains. Furthermore, the focus on ontology is usually on the class and concept structure, while knowledge graphs are mainly about the triples themselves. Thus several difficulties are added:

- *Diverse domain knowledge:* Since knowledge graphs are usually involving information about entities from different domains, the types of entities are usually extremely heterogeneous. It automatically leads to more complex heterogeneity issues.
- *Multi-linguality:* Furthermore, multi-lingual knowledge graphs are built, e.g., Wikidata, making data integration even more complex because entities and facts may stem from sources in different languages.
- *Algorithmic scalability:* Today's knowledge graphs are huge. For example, Wikidata contains more than one billion triples, about almost one hundred million entities. Since matching problems often require a quadratic comparison of all entities of two knowledge graphs, many existing methods are hardly able to work on existing real-world knowledge graphs. Particularly novel machine learning techniques are often extremely resource-intensive. Thus, resolving heterogeneity issues on this scale is often difficult. It gets even more complex if more than two different knowledge graphs need to be integrated.
- *Unstructured data:* Similar to ontologies, knowledge graphs have no fixed structure for entities of the same type. Hence, entities of the same type within the same type often have totally different sets of relations and attributes. Particularly, large knowledge graphs contain long-tail entities with only a few attributes, while other entities have several thousand attributes. Performing an entity matching in such a setting is significantly more difficult than in small, domain-specific ontologies.
- *No standards:* Another critical point is that knowledge graphs do not necessarily stick to Semantic Web principles for ontologies. As an example, the largest open knowledge graph Wikidata is internally not represented as RDF. RDF-S and OWL are hardly used. As an example, entities are allowed to be an instance and a class at the same time [113]. However, some ontology matching tools require this information for creating alignments, as already criticized by Zhang et al. [124].

Overall, we have seen that heterogeneity issues in knowledge graphs are, on the one hand, similar to what we know from ontologies, but on the other hand, knowledge graphs are significantly larger, more diverse, and hardly stick to Semantic Web standards such as RDF-S and OWL.

Similar to ontologies we are confronted with four major problems: *entity heterogeneity*, *relation heterogeneity*, *class heterogeneity*, and *literal heterogeneity*. They are orthogonal to the types of heterogeneity that we have seen before: schematic and semantic heterogeneity. All four problems may either occur as intra-source matching problems (single-knowledge graph matching) or inter-source matching problems (multi-knowledge graph matching).

2.5.1 Entity Heterogeneity

Entity heterogeneity is concerned with finding entities within single or multiple knowledge graphs that refer to an identical real-world entity. For example, it is about finding instances of the real-world person Albert Einstein in multiple data sources are presented above. The problem is usually resolved by entity/instance matching systems.

Ferrara et al. have noted that instance matching is more complex than classical entity matching in classical relational databases [33]. It is mainly due to larger structural heterogeneity in ontologies and knowledge graphs. While in classical databases, the structure of entities of the same type is given, in ontologies and knowledge graphs, these entities usually have different attribute sets, even within the same knowledge graph. Therefore, matching and solving heterogeneities becomes much more complex and requires a different view on structural matching techniques.

As discussed before, Halpin et al. present more difficulties that come with instance matching. They show how identity is a strict criterion that may lead to intensive philosophical discussions [41]. For example, imagine a knowledge graph with general information about the person `Barack Obama` and another domain-specific knowledge graph about the `44th President of the United States`, which also was Obama. The view on whether these entities are identical or not is subjective. Practical analysis of Linked Open Data sources shows that many existing identity links for entities are incorrect [95]. In the end, this misuse of identifying links, whether intended or not, may lead to problems when performing querying or reasoning on the data.

2.5.2 Relation Heterogeneity

In knowledge graphs, relation heterogeneity is reflected by having multiple relations having identical semantics or being in a hyponym/hypernym relation. *Relation*, *property*, or *predicate matching* is similar to matching attributes in classical schema matching. For example, the semantics of the two relations `birthplace` and `bornIn` have the same real-world semantics since they both express the relation between a person and her birthplace. They may be used interchangeably in a knowledge graph without changing the meaning of the respective triples. They are synonyms. An additional problem of relation heterogeneity is hypernyms and hyponyms. For

example, relations `actor` and `movieActor` are in a hypernym relation, since every movie actor is also an actor.

Zhang et al. have noticed that relation heterogeneity is only rarely discussed in research, even though it is often a more complex problem [124]. Relation heterogeneity issues may cause incomplete query results as pointed out in [1]. Cheatam et al. have already noticed that finding relation alignment is much more complicated than finding class alignments [18]. Their analysis has shown that in the OAEI 2013 Conference track, systems performance for relation matching was three times worth (average F1-measure) compared to their performance in class matching. False positives often show highly similar relation labels but often differed in the domain or range they were used for. It implies that working on relation labels alone cannot achieve good results. Structural features, such as the domain and range, may help to improve the matching quality.

2.5.3 Class Heterogeneity

Similar to finding corresponding tables in different relational database schemas, in knowledge graphs, we may find classes with the same semantics. Finding class matchings is a classical task in knowledge graphs and ontologies. Given, for example, the class `Persons` and `People`, the task is to figure out that they have an identical meaning. When it comes to class heterogeneity, one is often not only interested in identifying equivalent classes but also in `subclassOf` relations between two classes. In general, the problem is similar to schema matching, and also existing solutions are comparable. However, classes in knowledge graphs (similar to ontologies) usually have no fixed schema, i.e., entities in the same class have different attributes. Many classes often are only populated with few instances. Thus, instance-based matching techniques are more problematic since not enough instances might be given to building a correct alignment. Furthermore, some knowledge graphs (e.g., Wikidata) do not properly distinguish between classes and instances at all, which might further complicate the matching problem.

2.5.4 Literal Heterogeneity

Similar to the previous heterogeneity issues, we might also be confronted with multiple literals having the same semantics. In contrast to entities, relations, and classes, however, literals hardly bear any structural information since they only occur in triples' object position. Sometimes maybe only a couple of times in the complete knowledge graph.

As an example, a literal may be a simple string, e.g., a name that has been written differently. *Albert Einstein* or *Einstein, Albert* both would be feasible labels for an entity. A literal may be a number that again may have multiple semantics. The number *1879* may be a distance in meters, a height of a building, and the melting point in degrees Celsius or Fahrenheit, or a year. Particularly a lack of semantics about the type or unit of a literal may to several problems in solving heterogeneities here.

2.6 Resolving Heterogeneity in Knowledge Graphs

Methods for solving heterogeneity issues in knowledge graphs are still a hot topic in research. Technological advances in natural language processing and machine learning have led to several new developments for tackling heterogeneities in knowledge graphs. Also, several existing matching technologies for ontology matching are directly applicable to integrating knowledge graphs and thus are described here as well.

We categorize the different approaches for solving heterogeneities in knowledge graphs along different dimensions. As already mentioned in the previous section, solutions may be categorized by their *matching goal*. Even though entity, predicate, class, and literal heterogeneities in practice all come together, many systems still only tackle a subset of these problems alone. Particularly prominent is entity matching because entities are the central component in knowledge graphs.

Another dimension already mentioned in the previous section is the *number of knowledge graphs* the methods work with. While some systems solve heterogeneities in single knowledge graphs (intra-source) by duplicate detection, most systems work on exactly two knowledge graphs (intra-source). Multi-knowledge graph scenarios with more than two knowledge graphs are rarely evaluated in research but are an important problem in practice.

Another exciting dimension for differentiating matching techniques is whether they need training data to work in a *supervised* fashion based on machine learning algorithms or whether they work fully unsupervised.

We additionally have added the fundamental *technology* used by the systems. Particularly ontology matching systems often combine multiple techniques.

A complete overview of the different systems categorized by different dimensions may be found in Table 2.3. Note that we survey a range of representative systems for dealing with knowledge graph and ontology heterogeneity. Due to the large number of systems available, we do not cover all of them.

2.6.1 Entity Matching

Entity heterogeneity in knowledge graphs and ontologies is usually considered as an entity matching problem between exactly two knowledge graphs. Today, there are various approaches for entity matching in knowledge graphs and ontologies: human-in-the-loop-based systems that use crowd workers or domain experts to create a matching. Older systems use unsupervised methods with structural and string similarity. However, the current trend is, similar to the entity matching trend in relational database research, supervised techniques built with state-of-the-art machine learning techniques only requiring small amounts of training data in the form of example alignments. Machine learning techniques are often favored because unsupervised techniques have shown quality issues, particularly when solving heterogeneities in specific domains. To use unsupervised techniques often an extensive parameter tuning for the specific domain is required to achieve good results.

Table 2.3: An overview of different matching and reconciliation systems and their relations along the different dimensions: Their *matching goal* ((E)ntity, (R)elation, (C)lass or (L)iteral), whether they use *supervision*, the *number of KG* that are matched and the *technology* which is used.

System's Name	Goal	Superv.	No. of KG	Technology	Reference
JAPE	E	yes	2	KG+attribute embedding	[104]
MTransE	E	yes	2	KG embedding	[22]
BootEA	E	yes	2	KG embedding	[105]
MRAEA	E	yes	2	CGN	[69]
-	E	yes	2	CGN	[118]
REA	E	yes	2	GNN+GAN	[87]
ITransE	E	yes	2	KG embedding	[126]
-	E	yes	2	KG+attribute embedding	[109]
SigMA	E	no	2	similarity propagation	[64]
RiMOM-IM	E	no	2	multiple techniques	[99]
Hike	E	no	2	crowdsourcing	[127]
-	E	no	k	clustering	[57]
-	R	no	1	clustering	[123, 124, 125]
-	R	no	1	frequent pattern mining	[1]
PropSim	R	no	2	string similarity	[18]
-	R	no	1	KG embedding	[55]
-	R	no	1	probability distributions	[23]
RuleAlign	R	no	1	rule mining	[58]
-	R	both	1	KG embedding+text	[51]
BLOOMS	C	no	2	string similarity	[49]
ROSA	R,C	no	2	rule mining	[36]
-	L	no	1	string similarity	[21]
RiMOM	E, R, C	no	2	combination	[65]
PARIS	E, R, C	no	2	probabilistic	[102]
LogMap	E, R, C	no	2	reasoning	[53]
DOME	E, R, C	no	2	word embedding	[45]
FCA-Map	E, R, C	no	2	FCA	[20]
POMap++	E, R, C	no	2	clustering+classification	[63]
AML	E, R, C	no	2	string metrics	[31, 32]

Supervised Entity Matching. However, only a few supervised approaches are evaluated for heterogeneous knowledge graphs directly, but in the particular case of cross-lingual entity matching [104]. Usually, a single knowledge graph in different languages and some example alignments are available to the matching systems. The goal is to perform an entity matching between the different language versions of the knowledge graph, a so-called cross-lingual matching. Even though most methods have been created and evaluated for cross-lingual matching, they also find alignments among heterogeneous knowledge graphs of the same language as demonstrated in [106].

Several methods for cross-lingual matching exist, which could be classified into translation-based models, based on knowledge graph embeddings, and into graph neural network-based approaches [87]. Translation-based methods usually work comparable to the idea of Chen et al. [22]. The technique is based on the idea of translation-based knowledge graph embeddings, high dimensional but latent vector representations of entities and relations, usually used for knowledge graph completion. Knowledge graph embeddings are trained on both knowledge graphs. The embedding

vectors of semantically equivalent entities/relations are aligned using the training data consisting of already aligned entities. In these aligned embeddings, the distance among entities, relations, and triples from both knowledge graphs is measured and used to align the entities. While previous approaches solely focused on structural embeddings, Sun et al. additionally include attribute information to improve the matching quality [104]. Since training data for matching tasks are often rare, an approach using a bootstrapping method has been presented [105]. If only few training data is available, already aligned entity pairs are used as additional training examples to improve the classifier.

In contrast to translation embedding-based methods, other methods stick to graph neural networks for cross-lingual entity matchings. Wang et al. note that previous translation-based methods all have the problem of embedding structural information of single knowledge graphs and the equivalence information of the two knowledge graphs into a single optimization problem [118]. To overcome this issue, they propose a convolutional graph neural network (CGN) to embed entities into a vector space and train the model such that equivalent entities are close to each other. The advantage is that structural differences in the two heterogeneous knowledge graphs do not affect the resulting quality as much as in traditional knowledge graph embedding models. Drawbacks in this CGN approach are tackled by adding additional meta relation-aware components [69]. The authors of this work show that the original implementation by Wang et al. does not correctly capture relation semantics and add the relation type, relation direction, and inverse relations. This additional information significantly improves previous approaches. Furthermore, they have shown how an external machine translation technique may be used for getting training alignments for entities by translating their natural language labels. This idea enables the method to work unsupervised. REA, another current system for supervised entity matchings, aims at being unsusceptible for labeling errors in the training data [87]. Similar to previous works, they build on a graph neural network. However, instead of using a CGN, they build a message passing architecture for graphs. This work's main contribution is to include an extra module for noise detection, which is trained via a Generative Adversarial Network (GAN). This module additionally detects erroneous training data with high quality, giving an additional boost to the overall matching quality.

Inspired by the approach for cross-lingual matchings by Chen et al. [22], Zhu et al. presented one of the first supervised approaches for aligning entities in heterogeneous knowledge graphs based on latent graph representations [126]. Similarly to the cross-lingual approach, they use knowledge graph embeddings (PTransE) and perform an iterative matching of the embeddings. Moreover, the method achieves a high quality on a synthetically build benchmark dataset for heterogeneous knowledge graph matching. Recent work for integrating heterogeneous knowledge graphs is adding character-based attribute embeddings to align the entities embeddings [109]. Instead of just relying on structural similarity metrics as regular knowledge graph embeddings, this additional string similarity component is important for aligning heterogeneous entities. Structural features alone may not be sufficient.

Unsupervised Entity Matching. In contrast to supervised methods, significantly fewer methods have been published on unsupervised entity matching recently. Most available methods for entity matching are complete knowledge graph matching systems which solve entity matchings together with relation and class matchings. Therefore most unsupervised entity matching systems are described in the knowledge graph matching section.

Matching systems usually perform a pairwise comparison of all entities of two knowledge graphs, measuring their similarity using structural and string metrics. Lacoste-Julien et al. point out that these existing solutions are not scalable for large knowledge graphs [64]. Their system SiGMa is a greedy-based approach that significantly boosts the efficiency of the matching algorithm while keeping high-quality results comparable to existing matching systems. RiMOM-IM [99] is also focused on the scalability of the matching system. However, they use a blocking technique based on predicate and object string features so that a complete pairwise comparison of instances is not necessary. Concerning the quality, RiMOM-IM outperforms all other participants in most tasks at the instance matching (IM) track of the OAEI benchmark. Yan et al. propose a crowd-enabled entity matching system called Hike [127] This work aims to automatically perform a first matching step so that as few crowd workers have to be asked as possible. In the first step, a blocking based on predicate similarity is performed. Automatic matching candidates have a large influence on other matches and are sent to crowd workers first to prevent substantial errors.

A detailed comparison of several supervised and unsupervised methods on the cross-lingual matching task can be found in [106]. They show that, indeed, classical unsupervised methods are often better than modern machine learning methods and show complementary results.

All entity matching tools presented here have in common that they work on precisely two different knowledge graphs to perform an alignment. Some of them are also able to perform entity reconciliation within a single knowledge graph. More on this topic can be found in Chapter 3 of this work.

2.6.2 Relation Matching

To the best of our knowledge, most available methods for relation matching are all fully unsupervised. Only a single supervised method is available: It detects hypernyms/hyponyms, while all other systems focus on equivalent relations. Some approaches are built to perform relation matchings within a single knowledge graph, and others are built for two. Like entity matching, only a few methods work purely on relations but are full knowledge graph matching systems described later.

Purely structural and unsupervised approaches for relation matching are either association rule mining-based [1, 58], knowledge graph embedding-based [55], clustering based [123, 124, 125], or string similarity-based [18].

To the best of our knowledge, one of the first systems for finding relation matchings in knowledge graphs was proposed by Zhang et al. [123, 124, 125]. The basic idea of the approach lies in measuring the triple overlap between different relations. Furthermore, the overlap in the subject entities and the relation cardinality is used to

create a similarity measure. All three measures together are then used in a clustering algorithm. The three papers evaluate the approach on different datasets. In [123] DBpedia and Syndice are used to identify synonymous relations. An evaluation for query expansion on DBpedia is shown in [125]. A larger evaluation on DBpedia is performed in the corresponding journal publication [124]. PropSim [18] is relying on string similarity of the labels, but also the similarity of the domain and range class of the relation to perform an alignment.

In contrast to these multi-knowledge graph matching systems, some systems exist to work in a single-knowledge graph environment to identify synonymous relations. In [1], association rule mining is used to identify relations often having the same object entities but rarely having similar subject entities. This approach fails if relations do not share any entities at all. To overcome this problem, we propose two techniques that are further discussed in Chapter 4. One is based on a Horn rule mining-based approach [58]. For each relation, a set of equivalent Horn rules is mined automatically. These Horn rule sets are then compared to the sets of other relations to identify similar relations. The other one is an embedding-based approach, which uses knowledge graph embeddings similar to several approaches for entity matching [55].

The only supervised method available is built to detect hypernyms and hyponyms in knowledge graphs [51]. In this work, unsupervised knowledge graph embeddings and supervised techniques based on textual features using distant supervision are compared.

2.6.3 Class Matching

Similar to relation matchings, few systems solely focus on class matchings. However, most systems built for knowledge graph matching also align classes. They are presented in the next section. A system only working on schema-level heterogeneities is BLOOMS [49]. It uses external knowledge from Wikipedia disambiguation pages and Wikipedia categories to construct trees of categories for each knowledge graph class. These trees are compared against each other to find an alignment.

2.6.4 Literal Matching/Canonicalization

Aligning literals in knowledge graphs is rare. To the best of our knowledge, only a single work on knowledge graphs is dealing with literal canonicalization [21]. Literals alone lack valuable semantics: They cannot be appropriately typed. Making statements in triple forms about literals is not possible. To overcome this problem, the work's goal is to provide semantic type information for literals and to map them to entities. In a first step, candidate classes for the literal are extracted. A binary classifier for semantic typing is trained on heuristically generated training examples. In a final step, the predicted class and string similarity measures map each literal to an entity.

2.6.5 Knowledge Graph Matching

The techniques presented so far focus on a single problem of data heterogeneity (entities, relations, classes, or literals), even though all problems usually go hand in hand. A large part of systems, usually coming from the Semantic Web community, resolves all kinds of different semantic heterogeneities in knowledge graphs. Most of these systems have been published at the Ontology Matching Workshop at ISWC and evaluated with the OAEI benchmark.

RiMOM is one of the early and long-standing participants at the OAEI benchmark [65]. It dynamically combines several similarity metrics. They are starting with a hand-tailored linguistic similarity measure. Later structural similarity measures are used to perform the final matching. PARIS is a probabilistic system using working in an iterative fashion [102]. It starts with a simple literal similarity function to obtain initial equivalence candidates for entities. These candidates are used to compute candidate correspondences for relations. These two steps are performed iteratively until convergence. In a final step, the class correspondences are computed based on prior computations. In contrast to previous matching systems, LogMap also uses logical reasoning for performing a matching [53].

Additionally, we sketch state-of-the-art systems that have been presented at the recent OAEI 2019 benchmark in the knowledge graph track [47]. The approach showing the best recall is the first matching approach based on formal concept analysis [17, 20]. It uses string similarity metrics to match entities which then are used to build a relation-based formal context. A mapping is then derived from the nodes of the respective formal concept lattice. Another approach at the knowledge graph track is based on simple string metrics, string metrics based on word embeddings, and some type/cardinality filtering steps achieving high-quality matching results [45, 46]. AgreementMakerLight (AML) combines several string metrics for creating an alignment. [31, 32] POMap++ first performs clustering to perform a blocking into smaller sub problems [63]. In the next step, a classifier for the alignments is trained for each of the clusters separately. The training data is automatically generated by cross-referencing entities with external knowledge.

ROSA is a rule mining system based on the Horn rule mining system for knowledge graphs AMIE+ [36]. It requires already aligned entities to find a matching for relations and classes. Eight different rules which express complex correspondences between two knowledge graphs are mined automatically. Since this approach solely relies on already aligned entities, it may miss out on entity heterogeneities.

2.7 Conclusion and Open Problems

Throughout this chapter, we have presented a detailed introduction to representation heterogeneity in databases, ontologies, and knowledge graphs. We presented different classification systems for heterogeneity issues. We have shown that most problems that we have in knowledge graphs are either *schematic heterogeneity* and *semantic heterogeneity* issues. These heterogeneity issues lead to two typical problems: *Schema Matching* and *Entity Matching*. These problems are found in classical relational databases, ontologies, and large heterogeneous knowledge graphs. Furthermore, we

have classified the matching problems along with their goals into entity, relation, literal, and class matching.

Even though the heterogeneity issues in knowledge graphs are similar to what we already have seen in ontologies and the Semantic Web, knowledge graphs come with problems on a different scale: (1) Knowledge graphs are from various domains, (2) possibly even from multiple languages, (3) they are huge and often comprise several hundred million triples, (4) they are extremely unstructured, and (5) do often not use Semantic Web standards.

The heterogeneity issues that need to be resolved have become more complex. However, due to the advances in machine learning and the availability of large training data corpora, many new challenges are solved by novel automatic matching methods. We have seen that particularly in instance matching for knowledge graphs, novel graph neural network-based methods have become common. However, these novel approaches seem not to solve the heterogeneity issues but require large amounts of training data [106].

One striking problem is the focus on unrealistic matching scenarios. Most existing papers evaluate their systems on small knowledge graphs in an inter-source matching scenario with exactly two knowledge graphs. Intra-source matching scenarios with a single knowledge graph are hardly considered at all. Also, inter-source matching scenarios with more than precisely two knowledge graphs are hardly evaluated.

Matching Multiple Knowledge Graphs. Existing benchmarks and evaluations are usually performing alignments on exactly two knowledge graphs. In entity matching, almost all systems we presented were evaluated on existing this pairwise matching. Also, all existing systems for ontology matching are restricted to this setting.

Obviously, pairwise matching can be extended to multi-knowledge graph matching scenarios by just performing an independent pairwise matching of all knowledge graphs. However this may lead to additional problems, since identity links, i.e. `owl:SameAs`, are transitive [48]. In Chapter 3, we give insight into multi-knowledge graph entity matching problems and present ideas on how to solve upcoming issues.

Representation Heterogeneity in Single Knowledge Graphs. Also, heterogeneity issues in a single knowledge graph scenario have hardly been discussed in previous research, as also noticed by Zhang et al. [124]. There exist some techniques for relation matchings in single knowledge graph scenarios. Other matching problems have hardly been discussed in single knowledge graph environments so far. Also, existing techniques for relation alignments usually require a large overlap between the relation's extensions to identify synonymous relations. Thus, they perform some kind of instance-based matching, similar to what is often done in multi-knowledge graph settings. This requirement often is not fulfilled in single knowledge graphs. Chapter 4 discusses how to solve existing problems in detecting synonymous relations in a single knowledge graph scenario.

Implicit Knowledge Storage. Instead of explicitly integrating data from different sources as performed for decades on various types of databases, novel natural

language processing techniques offer the possibility to implicitly store knowledge in unstructured form [108]. Thus, we perform querying on unintegrated, heterogeneous data directly using neural language models. In Chapter 5, we further discuss the idea of how novel language models and knowledge graphs are combined to improve knowledge graph querying.

3

Transitivity Issues in Instance Matching

In the previous chapter, we have discussed different kinds of heterogeneity issues prevalent in databases, ontologies, and real-world knowledge graphs. We have shown that particular entity heterogeneity is a research field that has gained a lot of attention over the last decades. The research has resulted in various techniques to resolve the problems in a manual, semi-automatic, or fully automatic fashion.

Particularly when it comes to large heterogeneous real-world knowledge graphs usually storing entity-centric information about one hundred million entities, such as, e.g., in Wikidata, integrating multiple IRIs with the same real-world semantics has resulted in several instance matching systems. These systems are usually compared in benchmark datasets. First and foremost, the Instance Matching track by Ontology Alignment Evaluation Initiative co-located with the Semantic Web conference is an important venue to measure the performance of new techniques to integrate entities [1].

Usually, the benchmarks for instance matching involve triples describing entities from two different knowledge graphs. For example, triples about Albert Einstein from the two knowledge graphs Wikidata (`www.wikidata.org/entity/Q937`) and DBpedia (`www.dbpedia.org/resource/Albert_Einstein`). An instance matching system's goal would be to perform a pairwise matching of all entities between the two knowledge graphs. In our example, this would lead to a match from Q937 to Albert_Einstein. Usually, an `owl:SameAs` link is then used to state the identity of these to IRIs.

However, in real-world scenarios, matching only two knowledge graphs is usually only a part of the larger goal to integrated multiple (more than two) knowledge graphs. When matching multiple knowledge graphs, large connected components containing several instances representing the same real-world entities are created. If A is `owl:SameAs` B and B is `owl:SameAs` C, then A must be `owl:SameAs` C according to the `owl:SameAs` semantics, whether some instance matching system has actually

[1] `http://oaei.ontologymatching.org/`

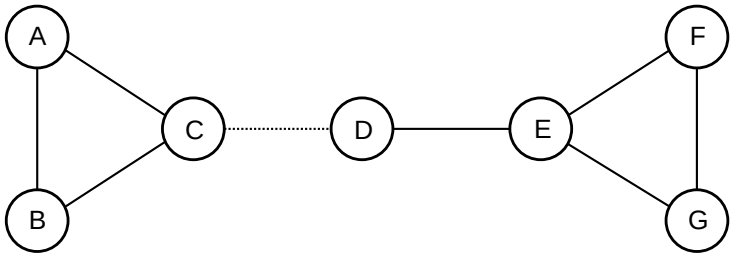

Figure 3.1: An `owl:SameAs` graph, where each node represents an instance from some knowledge graph, and the edges represent the created `owl:SameAs` links. The dotted line is an incorrect link.

declared A `owl:SameAs` C does not matter. Hence, a single incorrect identity may cause incorrect transitive links, propagating the error [48].

Example. Imagine instead of only two knowledge graphs, we want to integrate the entities from seven knowledge graphs. Instance matching systems find some `owl:SameAs` links between them as depicted in the *identity graph* in Figure 3.1.

We observe that seven correct links and only one incorrect link (dotted line) have been found by some instance matching system when performing a pairwise matching. A classical quality measure for instance matching would be precision, measuring the proportion of correct `owl:SameAs` links in contrast to all links that have been returned. In this case, the precision would be 87.5%. However, suppose we enforce the transitivity property of `owl:SameAs`. In that case, every two entities depicted in the graph in Figure 3.1 should be linked by an identity edge. Creating the transitive closure of such a class can lead to additional identity links. However, in our example graph, the transitive closure of the identity graph would consist of several links between non-identical instances since there was one single incorrect link. The precision of the original links and the derived transitive links drops to an overall precision of only 42.9%.

When dealing with several knowledge graphs where entity information is distributed over many sources, pairwise instance matching may result in chains of `owl:SameAs` links connecting instances, where single incorrect links may transitively lead to large errors. This phenomenon – similar to the game *Chinese Whispers* – is a problem for standard matching approaches, only focusing on two knowledge graphs at the same time. Subsequently, we analyze the problem of transitive `owl:SameAs` links when performing instance matching on several knowledge graphs and present several solutions.

Research Questions.

- *How do instance matching systems perform in a multi-knowledge graph scenario?*
- *How do we measure the matching quality of instance matching systems when multiple knowledge graphs are integrated?*
- *How do we improve the performance of arbitrary instance matching systems in multi-knowledge graph environments?*

Contribution. This chapter presents the problem of erroneous transitive `owl:SameAs` links when performing pairwise instance matching on multiple real-world knowledge graphs. To the best of our knowledge, we have published the first work, which explicitly analyzes the quality issues of the transitive closure of `owl:SameAs` identity graphs [48]. Additionally to this first analysis, we present four methods to deal with this so-called *Chinese Whispers* problem [57]. The presented methods improve the quality of any existing instance matching systems. Both publications, and therefore also this chapter, are based on ideas developed in my Master thesis [54].

Outline. This chapter starts with a section on related work in entity matching with a specific focus on transitivity of identity links in Section 3.1. In Section 3.2, we formalize the instance matching problem and basic notions to better understand the problem of incorrect transitive links. Our four methods to deal with the Chinese Whispers problem are described in Section 3.3. Finally, in Section 3.4, we provide insights on how we manually created a novel benchmark dataset for instance matching and evaluated the four presented methods on two state-of-the-art instance matching systems. In Section 3.5, a conclusion of our results is provided.

3.1 Related Work

Instance matching has been a heavily researched topic in the database, but also the Semantic Web community for several decades now, as already described in Chapter 2. Systems make use of a variety of different techniques to semi-automatically or automatically integrate entities from heterogeneous sources. But as shown in [48], most instance matching systems, also most of the systems we presented in Chapter 2 are only capable of matching exactly two knowledge graphs at the same time. Research about multi-data source entity matching is scarce. We try to give a short overview here.

3.1.1 Entity Heterogeneity in Relational Databases

In the database domain, matching problems integrating multiple instances of the same real-world entity are common due to a large body of work on duplicate detection within a single database, e.g., clean customer databases. The techniques can directly be applied to entity matching problems between multiple data sources (or knowledge graphs) as well.

Hassanzadeh et al. give a broad overview on how clustering techniques may be applied to duplicate detection problems by presenting the evaluation framework Stringer [43]. The input for Stringer is data sources with heterogeneous entity representations. Existing techniques for approximate joins are used to find initial matching candidates, so potentially identical pairs of entities. These entity pairs are fed into clustering algorithms to output classes of equivalent instances. Stringer compares several clustering algorithms on a variety of different datasets and provides a detailed analysis of the advantages and disadvantages of the algorithms. The partitioning algorithm, a single-pass clustering algorithm being used in various duplicate detection frameworks, simply computes the transitive closure of all connected instances leading to the Chinese Whispers problem described in the introduction of this chapter. Most of the other clustering techniques presented by Hassanzadeh et al. are inherently able to deal with the transitivity of `owl:SameAs` links and therefore prevent the Chinese Whispers problem. The Markov clustering algorithm shows particularly good results in many experiments. Inspired by these clustering results for duplicate detection in the database domain, in this chapter, we also employ clustering techniques to support matching transitivity problems in knowledge graphs.

3.1.2 Entity Heterogeneity in Knowledge Graphs

To the best of our knowledge, the first instance matching system for knowledge graphs, which was able to match multiple graphs at the same time, was LINDA [10]. LINDA is a map-reduced-based technique working with string similarity comparing n-grams of entity literals and a contextual similarity considering the neighboring entities. The matching algorithm works iteratively and starts matching entities with high literal similarity and puts them into the same equivalence class. Then the contextual similarity between different equivalence classes is computed, and equivalence classes might be joined. After several iterations, identical instances are within the same equivalence class, and `owl:SameAs` links are added.

Thus, in contrast to existing systems, the knowledge of transitivity of `owl:SameAs` links is used in the matching process of LINDA in a greedy fashion. Entities put into the same equivalence class at an early stage of the matching process cannot be split up later in the process if additional instances in the same equivalence class would indicate that splitting up the equivalence class may improve the overall matching quality. Unlike the methods presented in this chapter, the full potential of the transitivity property of `owl:SameAs` is therefore not used. Unfortunately, the implementation of LINDA was not openly available. A comparison to our methods is therefore not available.

A more recent method for solving entity heterogeneity in knowledge graphs was presented by Raad et al. [93, 94]. They use community detection methods to identify incorrect identity links in a large `owl:SameAs` corpus with more than 500 million links. In contrast to our work, they try to detect erroneous links, not on the output of instance matching systems (including their confidence values), but on existing `owl:SameAs` links in the Linked Open Data cloud. The authors rely on using the Louvain community detection algorithm to identify dense groups of instances that

may represent the same real-world entity. Louvain community detection computes an error rate for every `owl:SameAs` link in the network. Links with a high error rate may be incorrect `owl:SameAs` links and are removed, while the ones with low error rate are usually correct links since they are in the same community.

Applying the community detection algorithm on the identity graphs on the existing 500 million identity links led to around 55.6 million communities with an average of 3 entities per community. Particularly large groups with more than 165,000 entities are split into significantly smaller communities by their community detection algorithm. Therefore the quality of the transitive closure of `owl:SameAs` links is improved significantly. A manual analysis of the large communities has shown that communities indeed often group related entities, but not necessarily identical ones. The experiments show that after applying community detection, links with a low error rate show high quality, while links with a high error rate are incorrect in around 30% of the cases. Links with a high error rate (around 1.0) are incorrect in 40% of the cases but higher in the huge connected components. To summarize their findings: the analysis shows that community detection methods are helpful to identify incorrect `owl:SameAs` links to improve the quality of the closure of the identity networks from 61% to 91%. Since this work has been published only very recently, we could not perform a comparison. However, the goal of the presented technique is different from ours since it lies in an improvement on existing `owl:SameAs` networks and not in the improvement of instance matching systems directly.

3.1.3 owl:SameAs Networks

Analysis of existing `owl:SameAs` links in the Linked Open Data cloud provide a good insight into the problems of entity heterogeneity in existing knowledge graphs.

One of the first analyses has been performed by Ding et al., including almost 7 million resources, connected by almost 9 million `owl:SameAs` links [26]. Overall, the network consists of almost 3 million connected components with an average size of 2.4 resources per component. Large components usually are in the shape of a star, having a single central entity. Only around 41 components have a couple of hundred entities; the largest component contains around 5000 entities.

Beek et al. have performed a larger analysis of around 500 million existing `owl:SameAs` links in the Linked Open Data cloud and provide the dataset in a compact form [5]. Overall, the dataset consists of almost 180 million distinct subjects/objects, with a majority (97.4%) being IRIs, some being literals or blank nodes. The `owl:SameAs` network contains around 50 million connected components with a size larger than one node. 64% have a size of 2. The largest one has a size of 177,794. A closure, including the transitive, reflexive and symmetric closure of the `owl:SameAs` graph, leads to 35,201,120,188 identity links. However, around 90% of these links are linked to the largest connected component and may contain a massive amount of incorrect links. Exactly these incorrect transitive links in large components are what we focus on in this work to improve instance matching systems.

3.2 Preliminaries

As a first step, we introduce some formal definitions of the instance matching problem and the problem of Chinese Whispers. Furthermore, we introduce a new metric to assess the quality of matchings considering transitive matchings.

In this chapter, we work on the *instance matching problem* which is about creating new `owl:SameAs` links between multiple knowledge graph's entities. The knowledge graphs are usually provided in the form of multiple RDF triples with entities being represented as IRIs.

Definition 1 (Instance Matching Problem)**.** The Instance Matching Problem is defined for multiple knowledge graphs $KG_1, ..., KG_n$ with their respective sets of entities $E_1, ..., E_n$. Instance matching is defined for the set of all entities $E = E_1 \cup ... \cup E_n$: The goal for instance matching is to find a partition $P = P_1, ...P_m$ with $P_i \subseteq E$, such that every entity is contained in exactly one partition P_i. Furthermore, every P_i should only contain entities describing the same real-world object.

Usually, instance matching systems compute similarity values between each pair of entities. For $e_i, e_j \in E$ a similarity value $sim(e_i, e_j)$ between 0 and 1 is computed. High similarity values indicate that the corresponding instances are likely to represent the same real-world entity. Low values are a strong indication that they are dissimilar. Similarity is reflexive and symmetric. Unlike the identity relation `owl:SameAs`, similarity is not transitive: small differences among two similar entities may end up in a long chain where every two neighboring entities have high similarity, but the two endpoints are extremely dissimilar. As an example for this phenomena, we employ the well-known example by Luce [68]: Consider 401 cups of coffees, containing $(1 + \frac{i}{100})x$ for $i = 0, \ldots, 400$ grams of sugar, with x being the weight of one cube. The amount of sugar in cup i and $i + 1$ is nearly identical. However, coffee cup 0 has only one cube of sugar, whereas coffee cup number 400 almost only consists of sugar.

To get from this non-transitive similarity values to transitive `owl:SameAs` links, a matching function is employed, such that each similarity value is either assigned a *true* value or a *false* value.

Definition 2 (Matching Function)**.** Given a threshold $\Gamma \in \mathbb{R}$ and a similarity value $(sim(e_i, e_j))$ for two arbitrary entities e_i and e_j, a matching function is defined as $match : E \times E \Rightarrow \{true, false\}$ with

$$match(e_i, e_j) = \begin{cases} true, & \text{for } sim(e_i, e_j) > \Gamma \\ false, & \text{for } sim(e_i, e_j) \leq \Gamma \end{cases}$$

Entity pairs which are assigned *true* are considered to represent the same real-world entity and therefore are connected by a `owl:SameAs` link.

Example. We extend the example from the introduction by similarity values from an instance matching system as depicted in Figure 3.2. With a threshold of $\Gamma = 0.70$, the matching function creates 7 `owl:SameAs` links. A matching link between node D and E is not created. The identity graph is now consisting of two connected components. Entities within the same component are supposed to be identical.

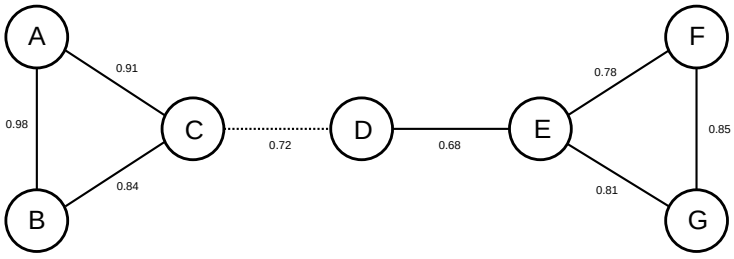

Figure 3.2: The graph from the introduction of this chapter together with similarity values from an instance matching system.

Our example shows that entities within the same component of the identity graph should be identical due to the transitive nature of `owl:SameAs`. Therefore, each connected component is also called an *equivalence class*.

Definition 3 (Equivalence Class)**.** An equivalence class of instance matching results is considered as a connected graph $G = (V, M)$ where $V \subseteq E$ is a set of entities representing the vertices of a graph and $M = \{m_1, m_2, \ldots, m_p\} \subseteq V \times V$ is a set of `owl:SameAs` links between the entities. The graph representation of an equivalence class is called an *identity graph*.

The entities within an equivalence class are defined as being identical in a pairwise fashion. The set of all identity links, i.e., edges between every two entities of the equivalence class, is called the *transitive closure* of the equivalence class.

Example. Let us now investigate the quality of the instance matching result to some instance matching problem. We have a closer look at the example identity graph in Figure 3.2. We pick a threshold of 0.60 for this example.

As already discussed in the introduction, the quality of the eight links is 87.5%. The transitive closure would contain all edges between the seven instances. Hence additional 13 `owl:SameAs` links between every two pairs of instances. The correct links are created between D and F and D and G. The 11 other links are incorrect due to the one incorrect (dotted) link between C and D. The precision of the transitive closure is only 42.9%.

The problem of creating incorrect transitive links is not reflected by traditional quality metrics in instance matching. Therefore, we introduce two new metrics to measure the *end-to-end join quality* by assessing the quality of matching systems considering the transitive closure. These metrics are built on top of classical precision and recall. Similar to precision, we define the *end-to-end precision* of an instance matching result as the precision on the transitive closure of the matches. Analogously, we also define the *end-to-end recall* as the recall on the transitive closure. The end-to-end precision in our example is only 42.9%, while the end-to-end recall is 100%.

Both measures are suited to better measure the performance of instance matching systems because not only explicitly created links but also implicit transitive links are considered. Even though the recall of instance matching systems is also important, our methods mainly focus on achieving high end-to-end precision without influencing the end-to-end recall too much.

3.3 Overcoming Transitivity Problems

In the previous section, we have shown that, indeed, the end-to-end precision of instance matching results for more than two knowledge graphs is significantly affected by the quality of transitively added identity links. Therefore, the quality of instance matching systems in a realistic matching environment might be significantly worse than in a simple scenario with exactly two knowledge graphs.

Overcoming the quality problem of transitive links is achieved by identifying incorrect `owl:SameAs` relations that connect correct subcomponents of an identity graph: When we come back to the example presented in Figure 3.2, its original end-to-end precision was only at around 42.9%. The removal of incorrect links could prevent a large proportion of the incorrect transitive identity links, which would boost the overall quality of the resulting `owl:SameAs` links significantly. As an interesting addition, also the removal of a correct identity link may improve the end-to-end precision by preventing a large number of incorrect transitive links.

Example. If we go back to our example from Figure 3.2, removing the correct `owl:SameAs` link between the instance D and E would boost the overall end-to-end join quality to 66.7%, since several transitively incorrect links are also removed.

To overcome this problem, we propose four methods to identify incorrect `owl:SameAs` relations to boost the end-to-end precision of instance matching systems. As an input, each of our methods uses the output of an existing instance matching system, hence similarity values between entities.

In contrast to existing matching systems, our techniques make use of the transitivity of identity links and work on top of equivalence classes to break them up with structural graph measures:

1. The first approach is working on the output weights of an instance matching system to remove weak matching links.
2. A clustering-based approach is based on standard community detection techniques, usually used for social network analysis.
3. We use the clique concept from graph theory to identify fully connected subgraphs in the identity graphs.
4. Another graph clustering approach, which is identifying dense regions using random walks.

3.3.1 Weakest Links

The first approach is a simple baseline approach, which does not consider the structure of the identity graphs but only the similarity values from the instance matching

systems. Since the computed similarity values are reflecting the confidence of an instance matching systems that two entities express the same real-world entities, the *weakest link* approach tries to remove links with low similarity values. Concretely, for each equivalence class with at least three entities, identity links with the lowest similarity values are removed until the class is split into two or more subcomponents.

Example. In Figure 3.2, the link between D and E is removed so that the equivalence class is split into two subcomponents. In this case, the edge with the lowest similarity value was a correct edge, and only the link between C-D, with a slightly higher value, is incorrect. Unfortunately, this was not detected by the algorithm, but the end-to-end precision is still improved because several transitively incorrect edges are removed anyways. The resulting end-to-end precision is 0.67.

3.3.2 Edge Betweenness

The *edge betweenness* approach is a graph clustering approach, taking into account the structure of identity graphs together with the similarity values output by the instance matching systems. The key idea here is that a high number of identity links within a group of entities of the identity graph is a strong indication that these entities are representing the same real-world entity. Hence, identifying such groups within an equivalence class that is better intra-connected than inter-connected and removing every other link should improve the end-to-end precision.

The edge betweenness idea itself stems from the field of community detection in social network analysis as heavily researched by Girvan and Newman [38, 77]. The algorithm we chose is called the Girvan-Newman algorithm, which is able to find highly connected subcomponents in a graph by identifying links between these subcomponents. Hence, it is based on edge betweenness, a measure for links on how likely it is that they are between two highly connected communities.

The edge betweenness of each identity link is computed by counting the number of shortest paths in the identity graph that go through this link. If two highly connected subcomponents are interconnected by a single link, the shortest paths between the nodes of these two communities all have to go through the single inter-connecting link. Hence, its edge betweenness value is high. By removing the link with the highest edge betweenness is an equivalence class and then recomputing the edge betweenness values, a graph is partitioned into communities. We apply the edge betweenness approach to equivalence classes with low quality and remove links until the class is split into two or more components.

Example. In our example, the shortest paths between every two nodes are computed. Since the graph consists of two highly connected components (A,B,C) and (E,F,G), the shortest paths connecting these subcomponents have to pass through D. Hence, the two adjacent edges, C-D and D-E have the highest edge betweenness values. In this case, the edge D-E is detected by the algorithm and removed from the identity graph, partitioning the equivalence class into two subcomponents (A,B,C) and (D,E,F,G) similar to the weakest link approach, resulting into a similar end-to-end join quality of 0.67.

3.3.3 Clique

The third approach aims to achieve high precision instance matching results by being restrictive, only keeping fully connected subcomponents of the identity graphs. Similar to the previous approach, the idea is also based on graph clustering, in this case, on complete-link clustering. The *clique* approach identifies subcomponents where the pairwise similarity between all entities of this component have a matching link, i.e., are above the threshold Γ. Hence, all resulting subcomponents are fully connected identity graphs. In contrast to the previous approach, the clique approach removes significantly more edges, particularly for larger equivalence classes.

Example. For the example, 2 `owl:SameAs` links are removed from the example graph, such that it is decomposed into 3 different subcomponents: (A,B,C), (D) and (E,F,G). Overall, one incorrect between C and D, but also one correct link between D and E was removed. This restrictive approach leaves us with only correct links in the two identity graph components. In numbers, this results in an end-to-end join precision of 100%.

3.3.4 Markov Clustering

An approach that has already shown excellent results in the work of Hassanzadeh et al. for duplicate detection is Markov Clustering [43]. Markov Clustering (MCL) was proposed as a random walk-based clustering algorithm for weighted graphs [111]. Similar to the edge betweenness approach, the algorithm detects dense regions (clusters) in graphs without specifying the number of clusters as an input parameter.

MCL is based on simulating random walks on the identity graph. The intuition behind using random walks is that they tend to end in the same dense region of the graph they started. If starting multiple random walks in the identity graph, edges that frequently used by random walks should be within the same dense region/community.

More formally, the random walks are described by two operations on a stochastic matrix created from a weighted adjacency matrix (build from the similarity values of the identity graph) by adding self-loops to each node and normalizing the entries, such that each column sums up to 1. Now the random walks are simulated by alternating an expansion and inflation step. *Expansion* corresponds to squaring the matrix, which simulates a random walk step. Thus, expansion increases the probabilities of intra-cluster edges. *Inflation* corresponds to taking the entry wise power of the matrix and normalizing the resulting matrix to be stochastic again. Hence, high entries in the matrix are further boosted in the inflation step, whereas low values are damped. After alternating both steps for several iterations, the matrix converges, such that clusters are formed. The inflation power is an input parameter, which is used to influence the granularity of clusters. The higher the inflation, the larger the number of resulting clusters. In our experiments, we chose the inflation parameter of 8.0 since it has shown promising results.

Example. On our example, the initial adjacency matrix, maybe transformed into the following stochastic matrix.

$$\begin{pmatrix} 0 & .98 & .91 & 0 & 0 & 0 & 0 \\ .98 & 0 & .84 & 0 & 0 & 0 & 0 \\ .91 & .84 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .72 & 0 & .68 & 0 & 0 \\ 0 & 0 & 0 & .68 & 0 & .78 & .81 \\ 0 & 0 & 0 & 0 & .78 & 0 & .85 \\ 0 & 0 & 0 & 0 & .81 & .85 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} .341 & .350 & .269 & 0 & 0 & 0 & 0 \\ .341 & .350 & .249 & 0 & 0 & 0 & 0 \\ .317 & .300 & .269 & .340 & 0 & 0 & 0 \\ 0 & 0 & .213 & .340 & .221 & 0 & 0 \\ 0 & 0 & 0 & .321 & .263 & .315 & .323 \\ 0 & 0 & 0 & 0 & .253 & .343 & .339 \\ 0 & 0 & 0 & 0 & .263 & .343 & .339 \end{pmatrix}$$

After 10 iterations of expansion and inflation the matrix converges as follows:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Hence, Markov clustering on the example identity graph would identify three clusters: (A,B,C), (D), and (E,F,G). Thus, it would identify the three dense regions in the graph by removing one correct and one incorrect link from the identity graph, similar to the clique approach. Similarly to before, the end-to-end precision is 100%.

Overall, we could see that (at least in our example) all four methods are able to boost the end-to-end precision in identity graphs and therefore have the potential of improving existing instance matching systems in a multi-knowledge graph environment. Detailed results on how the end-to-end precision is improved by still guaranteeing high end-to-end recall values are presented in the next section.

3.4 Evaluation

In this section, we describe the extensive evaluation of our four approaches on two state-of-the-art instance matching systems. Since no existing benchmark dataset meets our requirements, consisting of more than two knowledge graphs, we also describe the creation of a new benchmark for instance matching systems, which is built from a variety of real-world knowledge graphs. First, we provide an analysis of the benchmark dataset and discuss its difficulties. Then we evaluate the instance matching system PARIS [102] and SLINT+ [79] and finally discuss the resulting end-to-end join quality when working with the four approaches.

3.4.1 Building a Benchmark Dataset

We create a new benchmark for instance matching systems on multiple knowledge graphs ranging from general to highly domain-specific knowledge graphs. We use

Table 3.1: List of existing link types for creating the benchmark.

Source	Target	Type of Link
DBpedia	YAGO	Wikipedia
DBpedia	Freebase	`owl:SameAs`
DBpedia	KEGG	`dbp:kegg`
DBpedia	DrugBank	Chemical Abstract Service `(CAS) Number`
DrugBank	KEGG	`drugbank_vocabulary:xref`
NYTimes	Freebase	`owl:SameAs`
NYTimes	DBpedia	`owl:SameAs`
LinkedMDB	Freebase	`foaf:page`

Freebase [11], DBpedia [4], YAGO [103], New York Times[2], LinkedMDB [44], Drugbank and KEGG (from Bio2RRDF[3]) consisting of around 110,000 entities from the domains person, movie, book, organization and drug. The first four domains may be seen as general knowledge, whereas the fifth domain, drug, contrasts with the highly specific biomedical domain.

The datasets were chosen to introduce several difficulties: (1) Drugs are highly specific and usually have only a few descriptive triples in the knowledge graphs, (2) books and movies share several similar attributes (3) a large number of movies are based on books. Hence they have similar titles and sometimes even the same authors, and (4) we chose data from 7 different knowledge graphs to provide the opportunity to analyze the Chinese Whispers phenomenon in practice.

As a first step, we extracted all entities from the seven knowledge graphs that belong to at least one of the five domains using `rdf:type` properties. Since we are interested in a small dataset with many `owl:SameAs` links, we were trying to maximize the overlap between the knowledge graphs so that we ordered the entities alphabetically by their label and only took the top entities, resulting in 110,000 entities and around 5.5 million triples.

For creating the gold standard, we first explored existing `owl:SameAs` links in the datasets as shown in Table 3.1. It initially provided us with around 19,000 potential identity links. To verify if the various link types indeed could be seen as identity links in the context of our dataset, we randomly sampled 200 links per link type and manually checked if the connected entities indeed represent the same real-world object. All decisions made in this verification step were clear cut, without any border cases, where it was unclear whether the respective link was a correct or incorrect identity link. Therefore, we assume that the probability of creating false-positive identity links in this step is close to zero.

In the next step, we removed instances that had duplicates within the same knowledge graphs. After removing these initial duplicates connected by some identity link, we also performed duplicate testing using the two instance matching systems SLINT+ and PARIS. That way, we could identify another 29 duplicate instances

[2]This dataset publication is not available anymore.

[3]`https://bio2rdf.org/`

within the data. After this duplicate detection step, each entity is supposed to occur at most once in all knowledge graphs.

To obtain identity links among the remaining pairs of knowledge graphs that were not covered by links from Table 3.1, we computed the transitive closure for the existing 19,000 links. This step leads to an additional 13,000 links. The correctness of these links again was manually inspected by taking a random sample of size 200. All of them have been identified as correct.

Up to now, the detected `owl:SameAs` links are highly probable to be correct. However, we still may have missing identity links that have not been covered yet. Since DBpedia and YAGO IRIs are both based on Wikipedia, the mapping between these two knowledge graphs is assumed to be complete already.

For the remaining knowledge graphs, we manually matched all instances that have not been linked yet. To reduce the workload, we performed a pairwise comparison of all entities which have not been linked only if two entities belong to the same domain. E.g., a movie instance cannot be `owl:SameAs` a person instance. Furthermore, for every domain and knowledge graph, we manually chose a set of attributes that were automatically compared to reduce the number of manual comparisons further. So we did not compare persons with totally different names or birth dates. Overall, this manual comparison gave us 123 additional `owl:SameAs` links.

To access the quality of the resulting benchmark, we again took a random sample of 900 out of the n links, which all were correct. After these careful matching steps, involving several extensive manual verification steps, the false negatives, and false positives should be minimal. Thus, we present the first multi-knowledge graph entity matching benchmark with seven real-world knowledge graphs [4].

3.4.2 Experimental Setup

With the help of the created benchmark dataset comprising seven different real-world knowledge graphs, we are now able to analyze the Chinese Whispers phenomenon for two state-of-the-art instance matching systems that have been openly available for our usage. Since instance matching systems are built to perform pairwise matchings, we used the systems to perform matchings between each two of the seven knowledge graphs from our benchmarks. Afterward, we apply the four methods we have presented in the previous section on the instance matching results and evaluate the differences in end-to-end precision and recall.

PARIS. PARIS (Probabilistic Alignment of Relations, instances, and Schema) is built to match all items of standard knowledge graphs in a joined fashion [102]. They start with measuring the similarity of entities based on the object's similarity of shared relations. Next, the similarity of relations is computed using their overlap of entities. Then the similarity computation of entities is updated again by taking into account the new information on similar relations. This process is alternated, updating entity and relation similarities after each step until the process converges. Afterward, a similar process is performed for class similarity using the overlap of entities.

[4] `http://www.ifis.cs.tu-bs.de/staff/jan-kalo/benchmark`

For the evaluation of PARIS in the original work, the datasets DBpedia, YAGO, and the movie database IMDB have been used. The precision of the resulting matching between DBpedia and YAGO was measured by around 90%, the similarity from YAGO to IMDB was 97%, and the precision of the matching between DBPedia and IMDB was even 100%.

SLINT+. The second baseline is SLINT+ by Nguyen et al. [78, 79]. Two knowledge graphs are matched as follows: (1) The importance of predicates is measured and (2) predicates are aligned. (3) Now, instance matching candidates are computed using a similarity heuristic, which is based on the overlap of objects in the entities' triples. (4) The final step computes similarity values between good candidates by comparing matching predicates considering the importance of predicates.

The original system SLINT has been evaluated on five datasets (DBpedia, Freebase, NYTimes, LinkedMDB, and Geonames) in nine experiments. SLINT is then compared to 3 other instance matching systems, achieving an average precision and recall of 97% outperforming several other systems [79]. The extension that we used is SLINT+. It was evaluated on the OAEI benchmark in 2013 in 5 experiments, achieving an F1-measure between 0.878 and 0.999 [78].

Our Experiments. Overall, we perform four kinds of experiments.

1. First, we further analyze the Chinese Whispers problem on our benchmark dataset, analyzing the classical precision and recall and the end-to-end precision and recall of the two systems (SLINT+ and PARIS).
2. In a second analysis, we group equivalence classes which are output by the systems by their size, to evaluate their end-to-end precision. We conjecture that large classes have a lower overall quality.
3. A similar analysis is performed by grouping the resulting equivalence classes by their clustering coefficient. The *clustering coefficient* is a measure for the density of a graph. It is computed by normalized local clustering coefficients for each node of an equivalence class. The local clustering coefficient is the ratio of the edges that exist in the neighborhood of some node in relation to the maximum number of edges that could exist. Both analyses have been performed for a similarity threshold of 0.5.
4. After these preliminary analyses, we apply our four approaches (weakest link, clique, edge betweenness, and Markov clustering) to improve the end-to-end precision in identity graphs. Here, we, similar to the first experiment, use SLINT+ and PARIS to perform pairwise matchings among the seven knowledge graphs and then apply each of the four approaches. Our results yield precision and recall for similarity thresholds from 0.0 up to 0.9.

3.4.3 Analysis

The instance matching system SLINT+ has returned 98,799 possible `owl:SameAs` links, PARIS returned 105,168 links. All these links have a similarity value assigned by the respective matching system larger than 0.0 and at most 1.0.

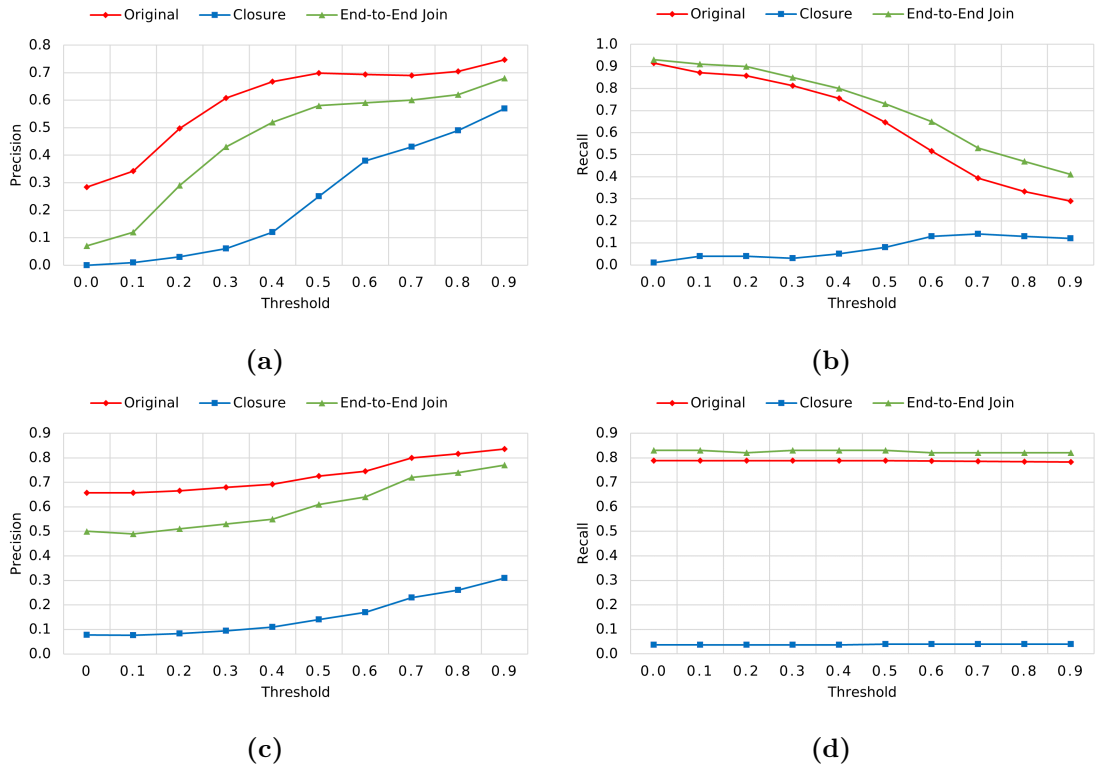

Figure 3.3: Precision and recall analysis for SLINT+ in (a) and (b), and for PARIS in (c) and (d).

End-to-End Precision and Recall of Instance Matching Systems. First, the precision and recall for the initial results of the instance matching systems, for the transitive closure only, and the combination of closure and initial results are analyzed. SLINT+ achieves a maximum precision 75% for a recall of 29% for a similarity threshold 0.9, cf. Figure 3.3 (a) and (b). At the threshold of 0.0, 28% precision and 91% recall are achieved. When we look at the results of the links that are added by building the transitive closure, their quality is significantly worse, achieving a maximum precision of 57% with a recall of 12% for the highest threshold. For a low similarity threshold, the recall of the transitive closure is only 1%, which may be explained by the original result of SLINT+ already comprising many correct transitive links. The transitive closure for SLINT+ has a size of over 300,000 additional links. The proportion of incorrect links is significant, as we observe by the difference in precision between end-to-end joins and the original precision.

The results for PARIS are depicted in Figure 3.3(c) and (d). It achieves a maximum precision of 84% at a threshold of 0.9 with a recall of 78%. The recall is hardly influenced by the similarity threshold. Hence, almost all correctly identified `owl:SameAs` links have a similarity value above 0.9. The transitive closure has a maximum precision of 31%, while the end-to-end precision is only slightly worse than the original precision. The small difference in the end-to-end precision is due to the small number of transitively added `owl:SameAs` links for PARIS.

These results show that the results on our benchmark are about 15% worse than the measured results from the OAEI benchmarks. The results for the end-to-end

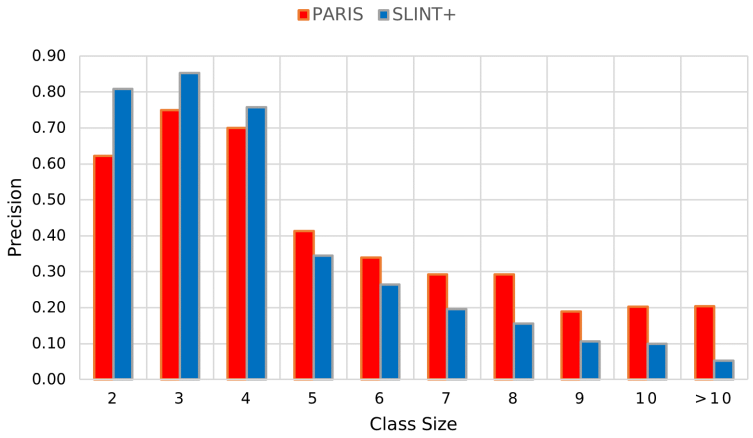

Figure 3.4: End-to-End precision for PARIS and SLINT+ per equivalence class, grouped by the class size of the equivalence classes.

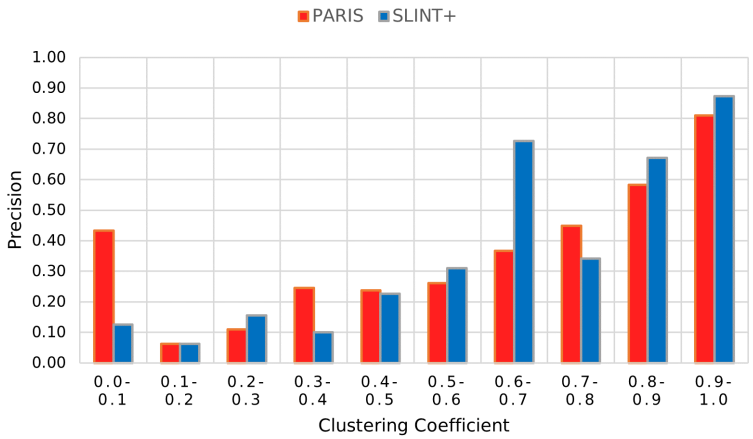

Figure 3.5: End-to-end precision for PARIS and SLINT+ per equivalence class, grouped by the clustering coefficient of the equivalence classes.

precision are even worse. Therefore, instance matching in a scenario with multiple knowledge graphs is far from being reliable.

Quality per Equivalence Class Size. Next, we measure the end-to-end precision for SLINT+ and PARIS categorized by the size of the equivalence class. The result of these experiments is depicted in Figure 3.4. Both systems achieve the highest precision for equivalence classes of size 3. Equivalence classes of size 2 have slightly lower precision. But in general, the precision for both systems is decreasing for larger equivalence classes. While SLINT+ has higher precision for classes until size 4, the precision for large classes is higher for PARIS. For classes with more than ten entities, SLINT+ has a precision below 5%, while PARIS still has a precision around 20%.

Quality per Equivalence Class Clustering Coefficient. In the second analysis, we have measured the end-to-end precision per equivalence class, grouping the

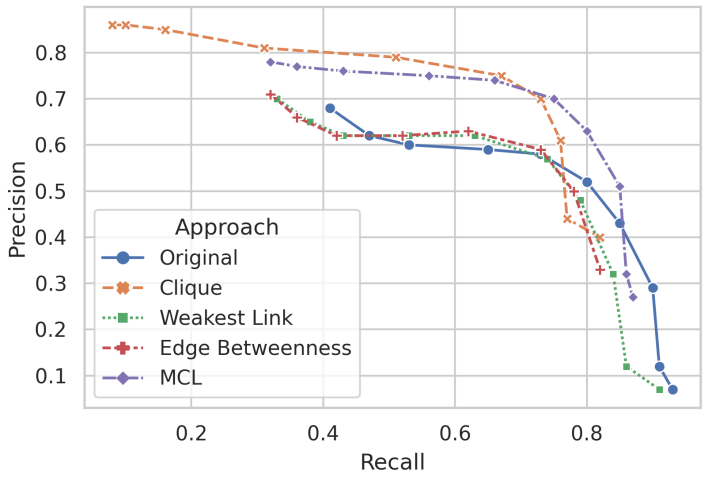

Figure 3.6: Precision and recall analysis for the four approaches *Weakest Link*, *Clique*, *Edge Betweenness* and *Markov Clustering (MCL)* on SLINT+.

equivalence classes by their clustering coefficient (cf. Figure 3.5). The precision of well-connected classes with a clustering coefficient above 0.9 is highest: For SLINT+ almost 90%, for PARIS 80%. With some exceptions, a lower clustering coefficient implies a lower precision. However, the precision for SLINT+ for a clustering coefficient between 0.6 and 0.7 is exceptionally high. It may be explained by SLINT+ having high precision for equivalence classes with three entities and only two links, which exactly fall into this category. Another interesting outlier is PARIS with a precision of over 40% for equivalence classes with a clustering coefficient between 0.0 and 0.1. Unfortunately, we have no further explanation for this observation.

In general, the experiments show that instance matching systems have a bad performance if equivalence classes are large and clustering coefficients are small. Whereas a large equivalence class often correlates with a small clustering coefficient.

3.4.4 Improving the End-to-End Quality

In our final evaluation, we measure the end-to-end quality (precision and recall) for PARIS and SLINT+ after applying the presented methods. We evaluate precision and recall for all four approaches independently.

The SLINT+ experimental results are presented in Figure 3.6. In blue, we observe the end-to-end join precision and recall of SLINT+ as discussed above. All four approaches increase the precision of the originally returned results and decrease the recall.

Let us first start to have a closer look at the weakest link approach. Maximum precision is at 70% in contrast to 68% for the original results, at the cost of 8% lower recall. The recall is often slightly worse. In summary, the approach shows few differences at all. Similar results are observed for edge betweenness, often showing no difference to the weakest link approach. However, its maximum recall is only 82%. As expected, clique achieves the highest precision results, with a maximum

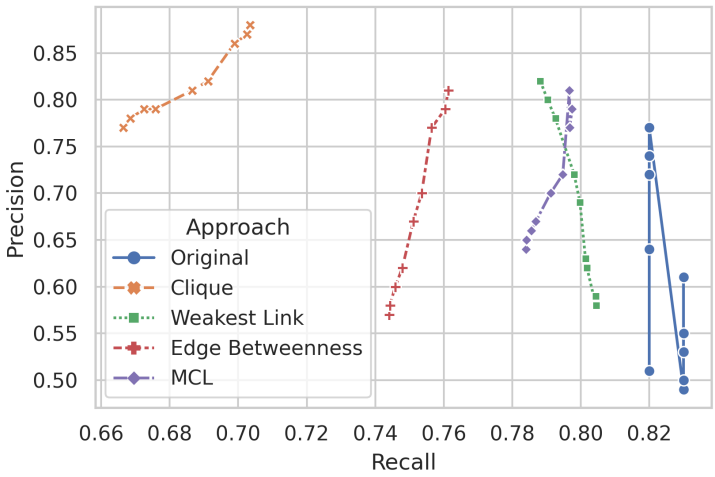

Figure 3.7: Precision and recall analysis for the four approaches *Weakest Link*, *Clique*, *Edge Betweenness* and *Markov Clustering (MCL)* on PARIS.

precision of 86%, boosting the original precision by almost 20%, but with a lower recall. For similar recall values, still, a precision increase of 12% is observed. Markov clustering (MCL) achieves high precision and still better recall than clique. Its maximum precision is 78%, for a recall of 32%, which is only slightly worse than clique. However, the maximum recall that is achieved by MCL is, on average higher than for the other approaches, with a maximum of 87%.

Overall, the results for SLINT+ show that the weakest link and edge betweenness perform worst and hardly improve the instance matching results. However, clique and MCL are both increasing the overall end-to-end join quality by more than 10% for similar recall values.

The precision and recall for PARIS are depicted in Figure 3.7. The precision-recall diagram for PARIS looks unusual. It is due to only small differences in recall that could be observed for different similarity thresholds, as observed for the blue line in the plot. The recall for all thresholds is either 82% or 83% with varying precision between 50% and 77%. Similar to before, the four approaches improve the precision for the cost of recall. Note that the scale for the x-axis and y-axis are different. Hence the losses in recall seem to be more significant than the gains in precision. The approach with the largest loss in recall is clique. However, a precision of 88% for a recall of 70% is achieved. Thus it increases the precision by 11%, with a loss in recall of 12%. Edge betweenness still achieves a recall of 76% with a precision of 81%. The weakest link approach achieves a top precision of 82%, increasing the original precision by 5%, with only losing 3% in recall. MCL keeps the recall high at 80%. On the downside, also the gains in precision for this clustering-based approach are only 4%.

In summary, all results trade precision for recall. In contrast to the results of the previous experiment on SLINT+, it is not possible to select a winner. All four approaches show a comparable trade-off between precision and recall. Clique being precision-oriented and MCL being recall-oriented.

Overall, our experiments have shown that all four techniques can be used to improve the end-to-end join quality by around 10%. The techniques have performed well for both instance matching systems. However, the improvements have been more significant for the instance matching system SLINT+.

3.5 Conclusion

Nowadays, there are many entity matching systems to deal with entity heterogeneity in knowledge graphs, as already discussed in Chapter 2. However, in this chapter, we have presented problems of existing instance matching systems that occur when matching more than two knowledge graphs at the same time. The quality problem that we have described is created by chains of `owl:SameAs` links, leading to a lower overall matching quality, called the *Chinese Whispers problem*. Since, to the best of our knowledge, no existing instance matching systems is able to completely deal with the Chinese Whispers problem, we first have introduced new quality metrics for instance matching systems in a multi-knowledge graph setting: end-to-end join precision and recall.

To the best of our knowledge, we presented the first methods that can directly improve the matching quality of arbitrary instance matching systems in multi-knowledge graph matching scenarios. The four approaches work on outputs of arbitrary instance matching systems and are able to boost the precision by over 10% with only to small losses in recall.

However, surveys of current entity matching systems show that state-of-the-art matching systems are still only working on two datasets at the same time and therefore ignoring the closure of identity links completely [106]. Hence, they ignore the Chinese Whispers problem completely, which may work out in artificial benchmarks or research paper evaluations but might be a big problem in real-world matching scenarios. Furthermore, they also ignore the valuable inside that is gained from transitive information and community structures to improve the matching quality.

For future work, it would be interesting to investigate more clustering algorithms for improving the end-to-end join quality in knowledge graphs. Our work has already shown significant differences between the four algorithms so that new approaches might be even better to solve the Chinese whispers problem. It would be interesting to integrate a technique for multi-knowledge graph matching directly into the instance matching system instead of making it a post-processing step. This integration could offer better possibilities to detect incorrect transitive links so that the precision might be boosted even more. It might be interesting to extend this work to relations and classes. We have already seen that instance matching problems in a multi-knowledge graph environment are becoming more difficult. A similar problem is to be expected in other scenarios as well.

4

Detecting Synonymous Relations

In the previous chapter, we were concerned with entity heterogeneity in a multi-knowledge graph scenario, causing problems due to transitive `owl:SameAs` links. In this chapter, we present a different type of heterogeneity: Heterogeneous relations within a single knowledge graph. Concretely, we analyze the problem of *synonymous relations*: relations have different IRI representations for the same real-world relation. In heterogeneous knowledge graphs, synonymous relations can be a massive problem. For example, in the real-world knowledge graph DBpedia, our analysis has shown that at least 27 different IRIs represent the `birthplace` relation. These synonymous relations include `bornIn` as a classic synonym, `birhtCity` with a typo in the IRI, or relations in different languages as `Lieu_de_naissance`. Thus, querying such knowledge graphs may lead to incomplete result sets if the relation synonyms are not detected and properly integrated with `owl:SameAs` links.

Example. As a short running example, the birthplace of scientists may be represented by multiple triples:

```
(Albert Einstein, bornIn, Ulm)
(Max Planck, birthPlace, Kiel)
(Gottfried Wilhelm Leibniz, lieu_de_naissance, Leipzig)
```

All three triples express the birthplace of famous German scientists. Einstein was born in Ulm, Planck in Kiel and Leibniz in Leipzig. However, three different relation representations are used: `bornIn`, `birthPlace` and `Lieu_de_naissance`. Thus, querying for famous German scientists' birthplaces may lead to incomplete and varying results depending on the query relation. As an example a user may pose the following SPARQL query:

```
SELECT ?person ?location
WHERE {
?person birthPlace ?location.
?person rdf:type 'Scientist'}
```

The query only returns Max Planck, leaving out the two other scientists from our example database. Here, detecting synonymous relations could help to return complete query results.

Even though a plethora of ontology matching systems exist that can perform a matching between two knowledge graphs, including an alignment of entities, relations, and classes (cf. Chapter 2), only a few systems are able to work in a single-knowledge graph scenario for performing duplicate detection [1, 124]. For matching relations, existing systems are usually based on computing an overlap between subject and object entities: Given two knowledge graphs with triples for `birthplace` in the one and `bornIn` in the second, the triples containing the relations usually share some person-birthplace pair. This overlap in their extension is used to perform an alignment. In a single knowledge graph scenario, this overlap often is empty, and thus, such a technique has problems in identifying all synonymous relations correctly. Furthermore, many matching systems heavily rely on string-based matching metrics on the IRIs of relations or their labels. However, relations' IRIs may be cryptic identifiers or have labels from multiple languages. We believe that both are problematic limitations that may lead to a large loss in recall when identifying synonymous relations within a large heterogeneous real-world knowledge graph. Therefore, in this chapter, we analyze structural methods for identifying synonymous relations in a single-knowledge graph scenario.

Research Questions.

- *Which techniques are suitable for measuring semantic similarity between knowledge graph relations?*
- *How do we use knowledge graph embeddings to identify synonymous relations in knowledge graphs?*
- *Are we able to detect synonyms in an interpretable fashion using Horn rules?*

Contribution. In this chapter, we show that synonymous relations within single knowledge graphs have hardly been tackled in previous works. We explore several novel approaches for resolving relation synonyms from large real-world knowledge graphs in a purely data-driven fashion. On the one hand, embedding-based approaches and, on the other hand, an interpretable approach using association-rule mining with Horn rules. All methods are not making any assumptions on the data and work independently of any string metric. Thus, they are applicable to most real-world knowledge graphs.

In large-scale experiments, we show that all our approaches detect synonymous relations with high precision. The contributions of this chapter are based on two publications: One was published at ISWC 2019 [55], the other one at ESWC 2020 [58].

Outline. This chapter covers multiple approaches for detecting synonymous relations in real-world knowledge graphs. We start by providing a broad overview for synonym detection in natural language, knowledge graphs, and open knowledge graphs in Section 4.1. In Section 4.2, we cover approaches based on knowledge

graph embeddings. We show how these existing techniques for knowledge graph completion may be employed for finding synonymous relations. We compare the embedding-based approaches to a novel rule-based approach in Section 4.3. Finally, in Section 4.4, we conclude the insights of this chapter on synonymous relation detection.

4.1 Related Work

In this section, we give a short overview of synonymous relations in different research areas. Starting with an overview of synonym detection in natural language, we discuss two techniques for synonym detection in knowledge graphs and one technique for hypernym detection in knowledge graphs. Furthermore, we give an overview of synonym detection in a related field using similar techniques as we do: canonicalization of open knowledge graphs.

4.1.1 Synonyms in Natural Language

Detecting synonyms with embeddings is a method known from natural language processing [119]. In NLP, words are often transferred into a vector representation by distributional models, such as Word2Vec [71] or GloVe [88]. Basically, synonym detection in natural language text is quite similar to our approach, but the data is different. For synonym detection, it is possible to either learn properties of synonyms with supervised learning on a set of known synonyms or in an unsupervised fashion similar to our approach. However, supervised approaches are much more common because large synonym repositories, such as WordNet [72] are available to serve as training data.

Since the labels of many relation synonyms in knowledge bases are also linguistic synonyms, combining NLP techniques with our method might be a possibility of improving the result quality if relation labels exist.

4.1.2 Synonym Detection in Knowledge Graphs

To the best of our knowledge, the research landscape for synonymous relation detection in knowledge graphs is scarce.

Abedjan and Naumann [1] argue that synonyms are problematic for querying since only partial results may be returned. To overcome this issue, they propose a query expansion using *synonymously used* relations. Two relations are synonymously used if one relation may replace another relation in a specific domain of interest. As an example, they use the relations `artist` and `starring`. These relations are not synonymous (as defined by us in the next section). In the context of movies, both relations are used synonymously since they describe the relationship between a movie and an actor. Some synonymously used relations are only synonymous in specific contexts and thus could be defined as hypernyms or hyponyms. As an example, `company` and `recordLabel` may be used synonymously for music, but an integration of both relations for the complete knowledge graph is not suitable. In this work, we do not use their definition for synonymously used relations but rely on

clear-cut synonymous relations that are generally valid as defined in the next section. Furthermore, restricting to clear-cut synonyms is easier to evaluate manually. Since the work by Abedjan and Naumann is the only available previous work at the point of publication of our work, we used it as a baseline.

The method that is proposed by Abedjan and Naumann is based on association rule mining. Overall, the proposed method consists of three steps: (1) All relation pairs with similar object ranges, i.e., a high proportion of overlapping object entities, are computed. This involves the computation of frequent itemsets using a minimum support value. (2) Relation pairs are filtered by the type of object entities. Similar to the first step, a high overlap in their type distribution is needed. (3) Synonymously used relations should not occur for the same subject entity. To achieve this goal, a reversed correlation coefficient is computed. (4) All relation pairs are ranked by the reversed correlation coefficient. Pairs with high rank are most likely to be synonymously used.

Abedjan and Naumann evaluate their approach on small and manually created datasets from DBpedia, Magnatune, and Govwild. The achieved precision values are extremely dependent on an input minimum support value and the dataset it is evaluated on. While on DBpedia in one setting, only a precision of 15% is achieved, on Magnatude, 100% precision is possible. In the end, the authors show that the second step (type filtering) of their method is not applicable for datasets with missing type information and is not improving the overall precision. Hence, in our evaluation, we left out the type filtering step and only implemented the first, third, and fourth steps.

Another approach for identifying synonymous relations was proposed by Zhang et al. [123, 124, 125]. The general idea of the approach presented in the three papers is similar to ours. It is a purely data-driven approach, which does not require any domain knowledge, and it is independent of the language since no string metrics are used for computing the similarity.

The matching approach identifies synonymous relations in a knowledge graph in three steps: (1) At first, a blocking technique is used to prevent a quadratic comparison between all pairs of relations. Only the similarity of relations having the same entity types in the subject entities are compared. (2) Three similarity measures are computed. This is a triple overlap, a subject overlap, and a cardinality comparison. The three measures are integrated into a single similarity measure. (3) Synonym groups of relations are formed by an agglomerative clustering using the similarity scores from the previous step. The output of the algorithm is clusters of relations, where each cluster contains only synonymous relations.

Overall, several experiments have been evaluated the performance of the technique in three different publications. In the first publication, experiments for evaluating the performance on synonymous relation detection have been performed on the datasets DBpedia and Syndice [123]. The precision in these small-scale experiments is over 70% for DBpedia at high recall values and over 85% for Syndice. In another publication, the approach has been extended to work for detecting synonymous relations to perform query expansion on DBpedia [125].

In the corresponding journal publication in [124], an extensive evaluation and discussion of the approach are presented. It involves a detailed evaluation of the performance of the approach on different DBpedia datasets and typical sources of errors.

Unfortunately, the implementation of the approach was not available for comparison. However, since the approach relies heavily on overlapping triples for the synonymous relations, we think the results should show the same weaknesses as our baseline by Abedjan and Naumann [1].

After the publication of our work, another approach for measuring semantic similarity between relations in knowledge graphs has been published [23]. Relation similarity is expressed over head-tail entity probability distributions using a feed-forward neural network. The similarity of two relations is computed by their Kullback-Leibler divergence.

As a baseline, the authors chose to compare to knowledge graph embedding approaches similar to what we propose in Section 4.2. The authors argue that their approach has two important advantages to knowledge graph embeddings: (1) Embeddings have a fixed dimension size for every relation and (2) the comparison of probability distributions has better interpretability than simply measuring the distance in the knowledge graph embeddings. Even though the evaluation of this work comprises several experiments on different datasets and a hand full of different applications, only two embedding baselines are used (TransE and DistMult). Both of them have only an average precision in our experiments presented in Section 4.2.

To give a better inside into the results of this work, we go into the details of their various experiments.

1. **Human Judgement:** In a first experiment, the similarity computations are compared to human judgments by computing the Spearman correlation on the judgments. While the presented approach shows a correlation of 0.63, DistMult achieves almost a 0.60 correlation.

2. **Synthetic Synonyms on Wikidata:** In this experiment, synonym detection is tested on synthetically created synonyms similar to our experiments in Section 4.2 and Section 4.3. In this experiment, the precision of their method is 65% at a recall of 50%. These are slightly better results than we show in our experiments on Wikidata in Section 4.2, but significantly worse results than our method in Section 4.3 However, since this work's experimental setup is slightly different, a detailed comparison to the results of our work is not possible.

3. **Synonym Detection on the ReVerb Dataset:** The ReVerb dataset is a large set of triples extracted from natural language text by an open information extraction method. Hence, the triples have no schema, and synonymous relations may have different IRIs. For their evaluation, the authors perform an approximation of precision and recall using a sampling method. The presented methods achieve a precision of around 30% for various recall values, while the knowledge graph embedding-based approaches show results below 10% precision.

4. **Relation Prediction on FB15K:** Relation prediction is about predicting new triples based on existing triples in a knowledge graph. For TransE [12], it is evaluated how relation similarity may be included into negative filtering for the embedding training to improve the prediction quality. Indeed, the quality for prediction on the FB15K knowledge graph completion dataset is improved by a tiny percentage.

5. **Relation Extraction on TACRED:** Similar to the relation prediction case, in this experiment, the relation similarity is integrated as an adaptive margin into the soft margin-loss of a relation extraction technique. The results show a slight improvement for one of the methods.

Overall, some of the experiments show that the presented methods are superior to knowledge graph embedding-based approaches. In other experiments, both methods seem to be on par. However, significantly fewer embedding methods have been evaluated, and the overall experimental setup is different. An interesting takeaway from this work is that synonymous relation detection can be used in various applications to improve the results of existing systems.

4.1.3 Hypernymous Relations in Knowledge Graphs

Additionally to synonymous relations, knowledge graphs also contain hypernymous and hyponymous relations. Two relations are in a hypernym relationship when one relation is entailing the other relation. As an example, `writer` is the hypernym of `screenwriter`. Analogously `screenwriter` is the hyponym of `writer`. Like synonyms, hypernyms may be misused and thus cause incomplete results when querying. For example, in a knowledge graph, `writer` could be used to relate a screenwriter to a movie, even though the more specific `screenwriter` relation should have been used. Using writer in this context is not incorrect but very unspecific. If some user would pose a query with the `screenwriter` relation, some results may not be returned due to this heterogeneity issue.

Jiang et al. presented an unsupervised and a supervised machine learning method that uses knowledge graph embeddings and textual embeddings to identify hypernyms and hyponyms in Wikidata [51]. First, a dataset based on Wikidata's property hierarchy is built. The authors evaluate the results from multiple features for an unsupervised method based on standard vector distances and a supervised method using a multi-layer perceptron. The knowledge graph embedding features are relation embeddings, similar to our work (cf. Section 4.2), and a combination of subject and object embeddings. For the textual features, sentences from Wikipedia are extracted in a distantly supervised fashion. Hence, sentences containing subject-object pairs for some relation are extracted and embedded using text embedding techniques. The last feature that was evaluated is a distribution of subject-object entity vectors to represent the respective relation.

In their experimental section, the knowledge graph embedding features were evaluated for the unsupervised method. The accuracy@1 is around 50%. The other features are used to train the supervised multi-layer perceptron method. The best results are achieved by the distribution-based method using textual features with an accuracy@1 of 70%.

The methods used in this work are significantly related to the methods that we present in the next section. Even though they are used for hypernym detection, they could be transferred to deal with synonyms as well. Unfortunately, the work was published after ours, so a comparison was not possible.

4.1.4 Open Knowledge Graph Canonicalization

The extreme case for heterogeneity in knowledge graphs is achieved by building schema-free open knowledge graphs using open information extraction. Open information extraction is concerned with the extraction of triples from natural language text without defining a schema (i.e., properties and classes) upfront manually. Due to the ambiguous nature of natural language, the resulting open knowledge graph is highly heterogeneous, requiring a so-called canonicalization procedure. Canonicalization integrates different entity and relation names when they describe the same real-world object. Thus, it is highly related to synonym detection in knowledge graphs. Concerning relations, several methods to detect synonyms in open knowledge graphs have been proposed. Two methods are particularly interesting to mention since they are comparable to our presented solutions.

Galárraga et al. propose a method for open knowledge graph canonicalization based on clustering and association rule mining. They evaluate their method on two standard open information extraction datasets: ReVerb and NELL [34].

As a first method, the authors focus on entity canonicalization. We do not go into the details here since the focus of this chapter is on relations. For identifying synonymous relations, the authors need already canonicalized subject and object entities of the open knowledge graph. As a next step, rule mining is applied to the semi-canonicalized triples to mine equivalence rules of the form $r \Leftrightarrow r'$ for high confidence values. These rules are found if r and r' have a high number of overlapping subject and object entities. As an additional feature, in this step, type constraints (if types are available) are enforced such that r and r' have to have the same domain and range types. In a final step, identity graphs of the relations are created using the equivalence rules with high confidence values. All relations within the same graph component are defined as synonyms. The general idea is comparable to what we propose in Section 4.3. However, our technique is more refined and does not need the respective relations to share any entities at all.

The evaluation of the relation canonicalization is performed only on the ReVerb dataset. Here, three different scenarios are evaluated: either using the entity clustering method presented in the paper or canonicalization using the Freebase knowledge graph. The latter method is evaluated with type information and without type information. Overall, the macro-averaged precision is between 81.3% and 94.6%. The recall is not evaluated.

CESI [112] is a pipeline of several methods to canonicalize an open knowledge graph. With regard to identifying relation synonyms, they mainly use two methods to create side information, a rule-based approach and relation extraction. Afterward, a knowledge graph embedding is used to cluster similar relations involving using this

information. The goal of the rule-based approach is similar to the previous method. They use Horn rule mining to find rules of the form $r \Rightarrow r'$ and $r' \Rightarrow r$ so that the equivalence of both relations is expressed. These rules are found when r and r' share several subject and object entities.

Additional side information is collected by relation extraction using distant supervision. In this step, subject-object pairs from an existing knowledge graph are linked to natural language text to find phrases describing a relation from the knowledge graph. Finally, a knowledge graph embedding is used to embed the relation phrases of the open knowledge graph into the embedding HoLE, initializing its weights with GloVe word embedding vectors. In this step, the side information created in the previous steps is included in the loss function of the embedding model. As the last step, a hierarchical clustering method identifies groups of relations that are returned as synonyms.

CESI is then evaluated on three datasets: Base, Ambiguous, and ReVerb45K. The three datasets were extracted from natural language text. For entity canonicalization, CESI was compared to several baselines, achieving an averaged precision of around 99%. For relation canonicalization, the method is compared against a rule-based approach inspired by the idea used for gathering side information before. Here, the macro-averaged precision is between 76.0% and 88.0%.

In contrast to our ideas on embedding-based synonym detection, CESI is evaluated on relational mentions from an open knowledge graph. Furthermore, in our work, we compare several embedding techniques and apply an outlier detection technique to identify synonyms. However, both methods share significant similarities with the methods presented in our work. The rule-based method was an inspiration used for the method we present in Section 4.3.

4.2 Knowledge Graph Embeddings for Finding Synonyms

Synonymous relations in knowledge graphs can cause several problems when working with knowledge graphs. We here present an unsupervised technique that uses the knowledge graph's structural features to identify relation synonyms with high quality to overcome this problem. In contrast to a variety of existing techniques, our goal was to create a technique that does not make any assumptions on the knowledge graph's data, does not require training data, and achieves high precision.

As described in the related work section, synonym detection has a long history in natural language processing and open knowledge graph canonicalization. A current trend in natural language processing identifies synonyms in texts by measuring their semantic similarity by computing a vector distance metric in some high-dimensional semantic space, .i.e. in word embeddings. In the present work, we transferred this idea to identifying synonymous relations using semantic embeddings of knowledge graphs. So-called *knowledge graph embeddings* have gained much popularity over the last years for knowledge graph completion, i.e., the prediction of new triples [81]. Knowledge graph embeddings are machine learning models that learn high-dimensional representations of entities and relations. Using vector arithmetics on these representations, they then predict unknown triples in a knowledge graph. Previous work has shown that these high-dimensional representations are also used to perform entity matchings [83]. Inspired by this idea, our work aims at using existing knowledge graph embedding techniques to identify synonymous relations.

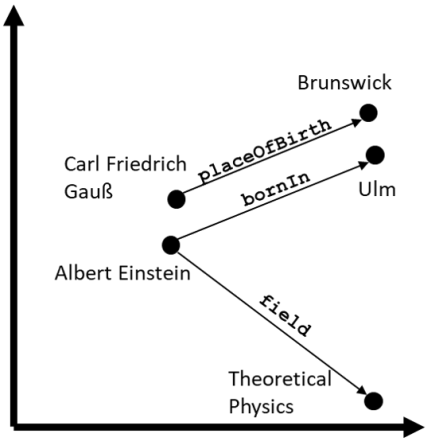

Figure 4.1: An example knowledge graph embedding about Albert Einstein and Carl-Friedrich Gauß showing three triples. The vectors for `placeOfBirth` and `bornIn` are parallel.

Example. We may obtain a vector representation for entities and a vector representation of relations for three triples about German scientists as depicted in Figure 4.1. Note that the subject and relation vector's sum is similar to the object vector, similar to the knowledge graph embedding TransE [12]. The similarity among relations is measured by their cosine similarity: Orthogonal vectors are dissimilar,

while parallel vectors have a high semantic similarity. In our case, the vectors for `birthPlace` and `bornIn` are similar. Hence, we conclude that they are synonymous relations.

In the following sections, we describe how to use eight knowledge graph embeddings with different relation representations to measure relations' semantic similarity. We use the models RESCAL [83], TransE [12], TransH [117], TransD [50], Complex [110], DistMult [121], ANALOGY [67], and HolE [82]. Furthermore, we present an outlier detection-based technique to identify synonymous relations with high quality. We evaluate the embeddings against a simple rule-based baseline on Freebase, DBpedia, and Wikidata on synthetic synonyms and by a large-scale manual evaluation. Our technique, the datasets, and the Code for reproducibility are published on GitHub [1]. The work that we present in this section has been published at ISWC 2019 [55].

4.2.1 Preliminaries

In this section, we work with knowledge graphs in RDF format consisting of subjects, relations, and objects. Our goal in this section is to introduce synonymous relations and then introduce different representation methods of relations that we may use to measure their semantic similarity.

A large knowledge graph contains several IRIs describing either entities or relations. Both are usually mapped to some real-world concept. As an example, the IRI `dbp:birthplace` may be mapped to the real-world relation that expresses the connection between some entity and the place it was born in. We define two relations r and r', with different IRIs ($r \neq r'$), as *synonymous*, when they describe the identical same real-world relation. It implies that in every triple s, r, o, the relation r may be replaced by r' into the triple s, r', o without changing the triple's semantics.

Similar to what we already did for entities in Chapter 3, our goal is to identify synonymous relations from a knowledge graph using a similarity measure between pairs of relations. It may be used to define a matching function similar to what we have already seen for entity heterogeneity in Chapter 3: The similarity of a pair of relations is measured using some similarity function. Depending on some similarity threshold Γ, we now either classify the pair as synonymous or not. Measuring the semantic similarity between relations is performed in different ways. We use knowledge graph embeddings to represent relations in a latent high-dimensional space. We compute classical similarity measures,i.e., vector and matrix similarity measures, between these representations.

4.2.1.1 Knowledge Graph Embeddings

Knowledge graph embeddings are techniques based on statistical relational learning [81, 115]. The goal of these methods is to create a latent representation of entities and relations that, on the one hand, describe the semantics of the objects and, on the other hand, is used to predict triples that are already in the knowledge graph, but

[1] `https://github.com/JanKalo/RelAlign`

also triples that are used to complete the knowledge graph. Recent work has shown that the entity representations of these embeddings may be used for measuring the semantic similarity between entities by applying vector metrics [83, 84]. In this work, we adapt this idea to relations.

The general training process of knowledge graph embeddings can be described as follows. We pick a form of representation for entities and relations. It may, for example, be a vector for entities and a vector for relations. Next, a goal function for the training process is needed. The training goal is learning to predict whether some s, r, o triple is in the training data or not. It needs to be represented in the form of a function that is used as an optimization goal. The goal function may be a simple translation between the entity vectors such that $s + r = o$ holds for all true triples of the knowledge graph and does not hold for all incorrect triples. During training, the vectors of relations and entities are optimized so that this goal function holds for as many true triples and as few incorrect triples as possible. After the training/optimization process, entity and relation vectors can predict new triples by vector arithmetics.

Example. The vector for `Albert Einstein` may be added to the vector of the `birthplace` relation.

$$v_{Einstein} + v_{birthplace} = v_o$$

The resulting vector v_o should now be similar to the vector of the actual birthplace of Einstein being `Ulm`.

From all models, we obtain a relation representation either in the form of a vector, as a matrix, or as a concatenation of several matrices. We measure the semantic similarity of the relations in a vector space using classical vector metrics. Since knowledge graph embeddings are currently not able to embed literal values or relations in triples with literal values, our method is restricted to relations between resources.

RESCAL [83] is one of the earliest embedding models for knowledge graphs. RESCAL is based on the idea of creating a low-rank tensor factorization of the original knowledge graph. The knowledge graph is represented as a three-way tensor $K = |E| \times |E| \times |R|$, having 1 at some position for valid and a 0 for invalid triples. The factorization is performed such that for each slice of the tensor, the following equation holds: $K_i = AR_iA^T$. The matrix A includes the latent representations of all entities, while R_i contains the latent representation of relation r_i. Hence, each relation is represented as a matrix of size $d \times d$, d being the number of dimensions, an input parameter. The details on how the factorization is performed as an optimization problem can be found in the original paper. We take the relation matrices as a representation for each relation.

TransE [12] is the first of a series of translation-based models for embedding knowledge graphs. The goal of all translation-based models is to optimize the distance of a latent entity and relation representations. For TransE, entities and relations are represented as a vector in some vector space. For some true triple (s, p, o) from the knowledge graph, the equation $o \approx s + p$ should hold. This equation

is used to optimize latent representations of entities and relations. The relations in TransE are represented as a vector of lengths d within the same space as the entities.

TransH [117] was proposed, since TransE has shown several weaknesses. Concretely, TransE could not deal with $one - to - many$, $many - to - many$, and reflexive relations properly. To overcome this problem, the goal function of TransE was adapted, such that the relation vectors lie in a separate space. The subject and object entity were first projected to a relation-specific hyperplane. On this hyperplane, the two entities are connected by a translation vector similar to what we have already seen in TransE. Overall, every relation consists of a translation vector and a normal vector used to project entities on the relation hyperplane. To represent an entity, we use the concatenation of these two vectors.

TransD [50] is another translational model that is supposed to overcome the drawbacks of TransE and TransH. In contrast to previous models, the goal of TransD was to put entity and relation vectors into separate vector spaces. Here, for each entity and each relation, two vectors are used for the representation: A standard vector and a projection vector. They are used such that it is possible to project entities and relations into a relation-specific vector space. We use the concatenation of both vectors as a representation to measure the similarity of the relations.

ComplEx [110] is a model based on complex numbers. The model's idea was to be fast and straightforward as the translation-based models but to be more expressive similar to RESCAL. To achieve this goal, the authors propose to use complex-valued embeddings using the Hermitian dot product. Hence, every entity and relation has a vector with a real-valued part, and an imaginary valued part. For our representation, we used the concatenation of the real and imaginary valued vectors for each relation.

DistMult [121] is seen as a simplification of RESCAL [115]. Like RESCAL, entities are vectors, but relations are also represented as a vector in the form of a diagonal matrix. As a result, the loss function is similar to RESCAL's, but with much fewer parameters. Therefore, it is less expressive than RESCAL, but faster to compute. We use the diagonal of each relation matrix as a vector for representing this relation.

HolE [82] combines ideas from factorization-based models, i.e. RESCAL with the simplicity of TransE (or DistMult). However, HolE does not have the problems that TransE has but is still simple to train. While RESCAL had large representations for entities and relations, HolE only uses vector representations for entities and relations. The loss function uses a circular correlation operation, which guarantees more expressiveness than TransE and DistMult, but still is scalable.

ANALOGY [67] is an embedding technique combining basic ideas of translation embeddings with analogical inference. The idea of analogies is that triple learning of one triple should profit from existing analog triples by applying additional constraints on the vector representations. Entities in ANALOGY are simple vectors. Relations

are mapping matrices that map a subject entity to the respective object entity for valid triples. The relation matrices are used for representation purposes.

4.2.2 Detecting Synonymous Relations

In the previous section, we have described how relations are represented in latent knowledge graph embeddings to measure their similarity. This section presents a method that takes knowledge graph embeddings as an input and outputs a list of synonymous relations in a purely unsupervised fashion. Hence, no synonym pairs are needed as training data.

Given some knowledge graph with its set of relations R, our method first creates a knowledge graph embedding from the original knowledge graph. Each relation is now either represented as a high-dimensional vector or a high dimensional matrix in this embedding. In the case of matrices, we concatenate the rows into a single vector. Hence, we obtain exactly one vector v_r for each relation $r \in R$.

Example. Imagine the knowledge graph contains the two triples `Albert_Einstein`, `birthplace`, `Ulm`) and (`Albert_Einstein`, `bornIn`, `Ulm`). A knowledge graph embedding is trained on these two and all other triples of the graph. For `birthplace`, we may obtain a vector $(0.3, 0.2, 0.3)$, while for `bornIn`, the vector $(0.55, 0.34, 0.72)$ was created. The cosine similarity among these vectors is 0.99, the cosine distance is $1 - 0.99 = 0.01$. A high cosine similarity implies that the two vectors are almost parallel. Therefore, the relations are also supposed to have high semantic similarity and are good candidates for being synonymous. Due to the high similarity of the two relation vectors, they are also used for triple prediction tasks. If we combine the vector presentation for `Albert_Einstein` with either one of the two vectors, the resulting vector is close to the vector of `Ulm`. A clear indicator that the vector similarity indeed is a good measure for semantic similarity.

We propose to work with a similarity measure and a distance measure. A distance of 0 or a similarity of 1 indicates identity: cosine similarity and L1-norm. *Cosine similarity* is defined as the angle between two vectors. $sim_{cos}(r, r') = \frac{r \cdot r'}{||r||||r'||}$. Cosine distance gives a value between -1 and 1. *L1-norm* is a distance metric also known as Manhattan distance. It is defined as $dist_{L1}(r, r') = \sum_{i=1}^{d} |r_i - r'_i|$, d being the number of dimensions of the embedding. In contrast to the cosine distance, the L1 metric is not restricted to a fixed range. It is a distance measure for the absolute difference of Cartesian coordinates between two vectors.

4.2.2.1 Classification

For identifying a list of synonymous relations, we need to perform a pairwise classification for each pair of relations from $R \times R$. As described in the preliminaries section, our goal for classification is to pick a threshold Γ, where all relation pairs with a higher vector similarity (or lower vector distance) are classified as synonyms. However, computing a global threshold for all relations is not suitable here because similarities between relations vary such that separating all synonyms from non-synonyms is not possible.

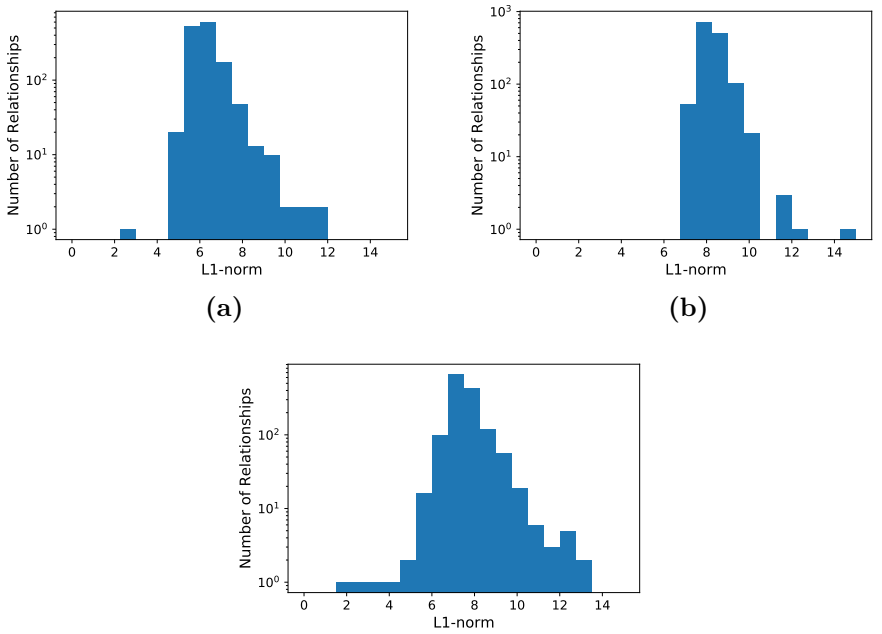

Figure 4.2: (a) A distance histogram from FB15K with the knowledge graph embedding TransE for the relation `award ceremony`, (b) `friend` and (c) `title`.

Instead, we have a look at distance histograms for each relation r by computing the L1-metric from relation r to all other relations in R. Three example histograms are plotted in Figure 4.2. For the histograms, we have counted the number of relations that have a certain L1-distance to the original relation r. Relations that are on the left-hand side of the diagram are usually similar or even synonymous. For the relation `award ceremony` in Figure 4.2 (a), the average distance is around 6, with a single outlier having a distance of around 2. Such outliers are usually synonym relations. Hence we would like to pick a threshold that separates the outlier from the rest of the distribution. When we look at the relation `friend` in Figure 4.2 (b), the average distance is around 7, no outlier on the left side of the distribution is found. In this precise case, no similar relation is found, and indeed the relation has no synonym in the dataset. For the relation `title` depicted in Figure 4.2 (c) it gets more difficult. The average distance is again around 7. However, the closest relation has a distance of only 2, while it is not a clear outlier. Indeed the relation has several synonyms, but they are difficult to separate from the rest of the distribution. Since all distributions are quite different from each other, a global threshold is not helpful here. Instead, we choose a distribution-dependent outlier detection that identifies outliers, such as in Figure 4.2 (a), but also synonyms as presented in Figure 4.2 (c).

To perform this outlier detection, we rely on distribution-based outlier detection, known as the *Z-score* [98]. The Z-score for some relation r and relation $r' \in R$ is defined as follows: $z_{r,r'} = \frac{dist(r,r') - \mu_r}{\sigma_r}$, μ_r being the arithmetic mean and σ_r the standard deviation. The Z-score measure detects outliers in the distance or similarity distribution based on the standard deviation of the arithmetic mean. Now, still, a global threshold for the classification of the Z-score is needed, which either is low

to achieve high-quality results or high to achieve recall-oriented results. We do not choose a fixed threshold here but evaluate the quality of our method for various thresholds resulting in precision-recall curves.

4.2.3 Evaluation

The following experiments are based on our previously published work [55]. Therefore the following experimental descriptions are a verbatim quote.

Overall, eight different knowledge graph embeddings on several real-world knowledge graphs are trained and compared to the method from Abedjan and Naumann from [1], which is used as a baseline. We employ the knowledge graph embeddings RESCAL, TransE, TransH, TransD, ComplEx, DistMult, HolE, and ANALOGY on Wikidata, Freebase, and DBpedia. Additional results and scripts for reproducing the results may all be found in our Github repository[2]. Our implementation of the knowledge graph embeddings is based on the framework OpenKE [42] which comprises nine knowledge graph embedding models. TransR [66] is excluded from the evaluation since it could not return any synonymous relations at all. The implementation of our classification, the evaluation scripts, and the baseline systems are in Python.

In this section, we want to evaluate synonym detection in a two-fold manner: (1) Experiments where we could evaluate precision and recall with synthetic synonyms, (2) but also a real-world scenario where we are not making any assumptions when generating synthetic synonyms.

Overall this resulted in three experiments:

1. We first experimented on a subset of Freebase (FB15K [12]) that is known to perform well for training knowledge graph embedding models. To measure recall and precision, synthetic synonymous relations are introduced into Freebase.
2. The second experiment is performed on synthetic synonyms in Wikidata. A knowledge graph that has, due to its size and sparseness, rarely been tested for knowledge graph embeddings. Since Wikidata's size is not suited for knowledge graph embeddings to be trained on, a sampling technique that still allows finding all synonymous relations is used.
3. The third experiment on DBpedia, a manual evaluation of the *Precision@k* instead of introducing synthetic synonymous, is performed. In contrast to Wikidata, DBpedia is much more heterogeneous because it comprises a larger number of relations. A measurement of the recall is not suitable here because no gold standard of synonymous relations is available. Building a gold standard would require manually checking millions of possible synonym pairs.

In a final discussion, a comparison of the various experiments is made. Cases where our technique could not identify synonymous relations are further discussed. The discussion also presents the advantages and disadvantages of the different models. It provides guidelines for choosing the right model for synonym detection.

[2]https://github.com/JanKalo/RelAlign

Baseline Based on Frequent Itemsets. In all experiments, the eight embedding models are compared to the baseline technique from [1]. Since no implementation is available for the baseline system for synonym detection, we re-implemented the *Range Content Filtering* and *Reversed Correlation Coefficient* as described in the paper. Further details on our Python implementation are available in our Github repository. However, the technique has a *minimum support* as an input parameter for the range content filtering step, which highly influences precision and recall. We performed a grid search on the minimum support to tune this parameter to achieve the highest F1 measure.

Synthetic Synonyms Generation. Synthetic synonyms are created by replacing relation IRIs with new (synthetic) IRIs in existing triples of the dataset. For example, we replace the triple (Albert Einstein, `award`, Nobel_Prize) with the triple (Albert Einstein, `award_synonym`, Nobel_Prize). `award` and `award_synonym` now have identical meaning and are treated as synonymous relations. Performing a proper relation matching requires the method to re-identify these synthetic synonyms from the knowledge graph. For the synthetic synonym generation, an assumption from [1] is used so that the baseline can perform synonym detection. Abedjan and Naumann assume that synonymous relations do not co-occur for the same subject entity. In the case of our Einstein example, all triples about his awards would either use `award` or `award_synonym`, but should not mix the two for the same entity. This assumption stems from the idea that entities and their triples are often inserted at once by the same person or from the same data source. Thus, synonymous relations for the same entity are rare. For the experiments with synthetic synonyms, we introduced exactly one synthetic relation for each relation that occurs in at least 2000 triples. We replaced it in 50% of the triples. The F1-measure for all methods, including the baseline method, decreases the more skewed the distribution is since it leads to some relations being extremely rare, which negatively influences the embedding representation of a relation. Results for the skewed distributions may also be found in our Github repository.

Sampling Method for Large Knowledge Graphs. Knowledge graph embedding training involves a lot of computational effort, which is why it should be performed on a fast GPU. Typical GPUs are restricted in their memory size, making it impossible to train models for complete knowledge graphs. Training embeddings for complete Wikidata on a CPU is technically possible but is around 10-100 times slower (i.e., several weeks) and thus prohibitive. To overcome this issue, we came up with a sampling technique covering all relations of a knowledge graph, but only a fraction of all triples. We randomly selected entities with all their triples to have similarly many triples per relation in our random sample. With this sampling method, we try to achieve that knowledge graph embeddings still work while having enough information about each relation. Its semantics is correctly mapped to the latent vector space.

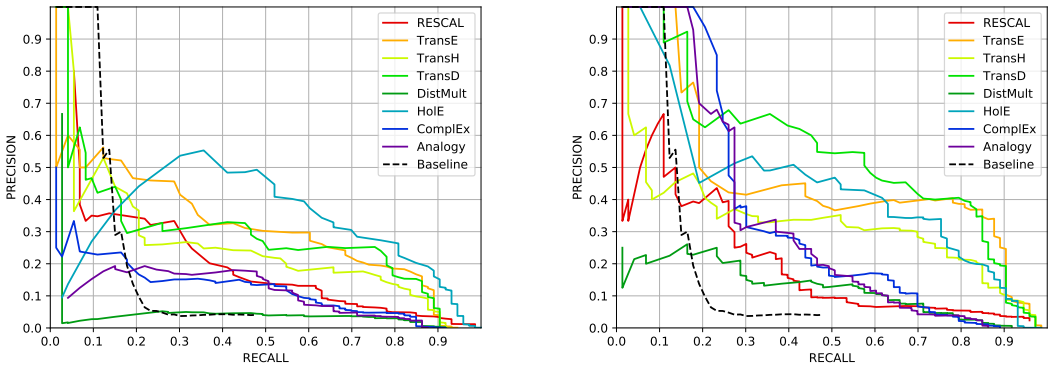

Figure 4.3: Precision-Recall-Curves for Synthetic Synonyms on Freebase. (a) Results with Cosine Similarity (b) Results with L1-Metric

4.2.3.1 Evaluation of Synthetic Synonyms in Freebase

In this experiment, we compared knowledge graph embedding-based synonym detection with the baseline system on a subset from Freebase (FB15K) that is usually used to evaluate knowledge graph embeddings on link prediction [12]. FB15K comprises 592,213 triples about 15k entities, using 1,345 different relation types. The dataset does not contain any literals, hence only triples where subject and object are resources. Originally, FB15K is a small part of Freebase that was chosen for link prediction because it comprises many triples per entity and lots of entities per relation. It has been shown that this dataset is particularly well suited for training knowledge graph embeddings, also leading to good results in other tasks, such as link prediction. Since no gold standard for the existing synonymous relations in FB15K is available, we have introduced synthetic synonyms. Overall, 74 synonymous relations have been added to FB15K.

The results of eight knowledge graph embeddings and the baseline are presented in Figure 4.3. The baseline achieves its highest precision of 1.0 at a recall of 0.11 but then drops to a precision of 0.05. For the minimum support of 0.02 leading to the best F1 measure, the recall never exceeds 0.5. It implies that more than 50% of the synonyms are never found. Lower minimum support also negatively influences the precision. Our knowledge graph embedding-based approach, on the other hand, is evaluated with cosine and L1 metric. For the cosine similarity in Figure 4.3 (a), the baseline performs best for low recall values. However, for a recall above 0.2, all models but DistMult perform better than the baseline approach. The result's quality is even better for most models with the L1 metric in (b). TransD is best in synonym detection, achieving 1.0 precision at a recall of 0.1 and still 0.4 precision at a recall of 0.8.

Knowledge graph embeddings in this dataset achieve high precision for low recall values and find many false-positive synonymous relations. These false positives are due to Freebase's fine granular modeling of relations, leading to many semantically similar relations that are not synonymous. Relations in Freebase are defined for each entity type separately, implying that each relation type is only used for a certain

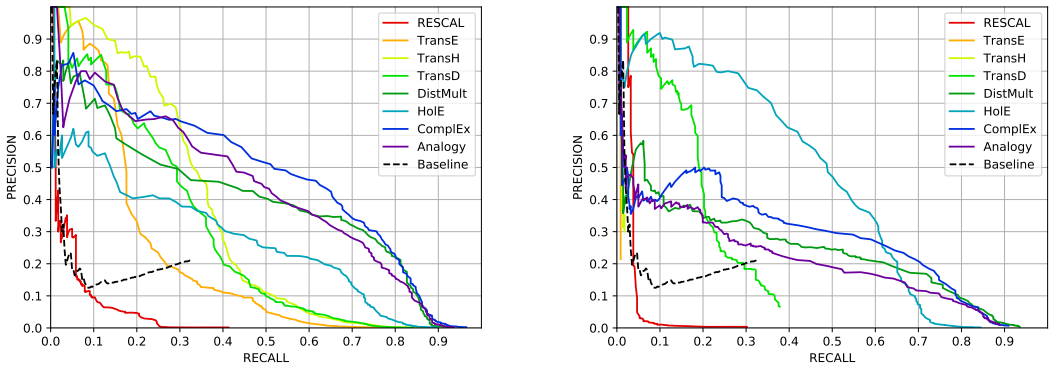

Figure 4.4: Precision-Recall-Curves for Synthetic Synonyms on Wikidata. (a) Results with Cosine Similarity (b) Results with L1-Metric

entity type. As an example, several `genre` relations are defined, depending on the class of the entities it is connected to. Differentiating `music_genre` from `film_genre` is quite difficult, but still possible with most embedding models. However, it gets even more complex: FB15K contains 33 different `currency` relations, all having slightly different semantics but similar extensions. It is hence a problem for data-driven synonym detection techniques when no background knowledge is given.

4.2.3.2 Synthetic Synonyms in Wikidata

The knowledge graph Wikidata is one of the fastest-growing knowledge graphs that are openly available today. Our Wikidata version is from 9-19-2018. In contrast to other knowledge graphs, the Wikidata community is investing much work into controlling its vocabulary. Therefore, it is supposed to be synonym-free, making it a great candidate for evaluating our method with synthetic synonyms. Due to its size, we did not train knowledge graph embeddings on the complete Wikidata knowledge graph, but on a sample that comprises 15,663,641 million triples, with 341 synthetic synonymous relations out of 1,797 relations.

The precision and recall curves for all eight models and the baseline are presented in Figure 4.4. The knowledge graph embedding-based approaches show a higher precision than the baseline for cosine similarity and L1-metric. Only RESCAL cannot hold up with any other system. The baseline starts with high precision but sharply decreases and ends at a precision of 0.2 at a recall of 0.3. For the optimally chosen minimum support, the baseline only returns one-third of all synonymous relations. ComplEx and HolE achieve the best classification results, outperforming the baseline by far. HolE has a precision of 0.75 at a recall of 0.3 and then is decreasing (cf. Figure 4.4 (a)). ComplEx, in contrast, is starting with a lower precision but still has a precision of over 0.5 at a recall of 0.5 (cf. Figure 4.4 (b)).

Training good knowledge graph embeddings on a knowledge graph that is as sparse as Wikidata leads to lower quality models in contrast to FB15K It also impairs the quality of synonym classification. However, Wikidata, in contrast to FB15K, does not contain highly similar relations that could be misjudged as false positives

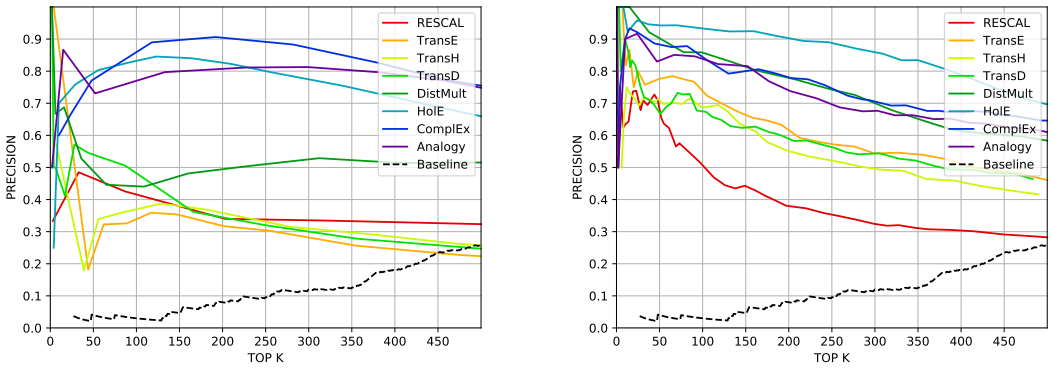

Figure 4.5: Manually evaluated Precision@k for Synonyms in DBPedia. (a) Results with Cosine Similarity (b) Results with L1-Metric

by the classification technique. These two factors even out each other leading to a comparable quality to FB15K from the previous experiment.

4.2.3.3 Finding Synonyms in DBpedia with Manual Evaluation

As a last experiment, we also want to show that our method identifies existing synonyms in a large-scale and heterogeneous knowledge graph. Therefore, we evaluate our method with all embedding models and the baseline on a sample of DBpedia-16-2010. Due to its size, again, a random sample similar to the procedure before is taken, resulting in a dataset with 12,664,192 triples and 15,654 distinct relations. For the manual evaluation on DBpedia, the annotator were supposed to evaluate relation pairs into *synonyms* and *non-synonyms*. To measure the task's difficulty, we first measured the inter-annotator agreement on a small sample of our dataset. We achieved an annotator agreement of over 0.90 for two independent raters, implying that they came to similar results. Due to this experiment and the dataset's size, we decided on only a single annotator for the manual evaluation. This manually-built dataset stems from the top 500 results for each embedding model and the baseline summing up to around 3600 relation pairs, of which 1100 have been classified as correct. The dataset is freely available online [3]. With this dataset we are able to obtain *Precision@k* values up to $k = 500$.

The results as Precision@k of our manual classification are presented in Figure 4.5. For the baseline approach in this experiment, we chose a minimum support that returns around 500 results to be comparable to the other results. Choosing lower minimum support would increase the number of returned results but decreases the precision. Compared to the other models, the baseline starts with a low precision for $k = 50$, with a steadily increasing precision of up to 0.25 at $k = 500$. Note that the baseline is never exceeding a precision of 0.3 with the chosen minimum support value. The unconventional behavior of the curve is due to the assumption synonymous relations never co-occur for the same subject. This assumption is not

[3] https://figshare.com/s/11d4af3169a0e6d2437b

true for DBpedia. The precision of our classification method on top of knowledge graph embeddings is showing higher precision for almost all models. HolE, ComplEx, and ANALOGY all show comparably high precision values for high k values. In contrast, the translation embedding models TransE, TransD, and TransH are quite weak to the earlier experiments. HolE with L1-metric in Figure 4.5 show the best results with a precision of 0.94 at $k = 50$ and still a precision of 0.7 at $k = 500$.

During the extensive manual evaluation of the models, we got a detailed insight into the advantages and disadvantages of the models on DBpedia. Widespread synonymous relations that can clearly be distinguished from others are clearly identified as synonyms. These are for example relations for `genre`, `almaMater`, `deathPlace`, `birthPlace` and `award`. Problematic, at least in DBpedia, are rarely used relations (`fuelSystem`, `drums`), relations with spelling errors in their label (`amaMater`, `birthPace`) and relations that are similar to others other existing relations (`club`, `youthteam`). Several other false positives stem from DBpedia containing relations that are automatically extracted from external data sources that should be integrated and reformulated. As an example, it imports an external baseball database by creating two relations for every row of a table: e.g. `stat1label`, `stat1value` for the first row and `stat2label`, `stat2value`. These false positives are not synonymous relations but problematic relations that should be reformulated.

In all three experiments, we have shown the advantages of our embedding-based classification method on various knowledge graphs. The baseline has been outperformed by almost all embedding techniques because it heavily relies on synonymous relations to share object entities. In contrast, knowledge graph embedding-based approaches detect synonyms even though they do not share any subject nor object entities. As an additional drawback, the baseline requires parameter tuning for the minimum support value.

We have seen that many synonymous relations are detected in knowledge graphs if they are frequently used. The semantics of rare relations can hardly be mapped to the knowledge graph embedding, hindering the data-driven synonym detection mechanism. All embedding models show varying qualities across the different datasets, with HolE showing consistently good if not the best results when choosing L1-metric. For most other models, L1-metric is also showing better results. Still, no model could identify all synonymous relations with high quality only based on the knowledge graph itself.

The fine-grained modeling of relations (as in Freebase and DBpedia) is often problematic since these relations may hardly be distinguished from real synonyms, even in our extensive manual evaluation. We observed that relation pairs that have been counted as false positives often are pairs of relations that are incredibly similar.

For example, `/education/university/local_tuition./.../currency` and `/education/university/domestic_tuition./.../currency` both are highly similar in their extension, however are, semantically speaking, slightly different. One is used for the currency of the tuition at universities for local students, the other one for domestic students. The semantics of local and domestic students might be very close but is not synonymous. We believe that these relations could be integrated.

However, detecting such a difference by a purely data-driven approach seems to be almost impossible.

4.2.4 Discussion

In this section, we have shown how relation representations from knowledge graph embeddings are used to identify synonymous relations in real-world knowledge graphs. We have performed experiments on Freebase, Wikidata, and DBpedia and showed embeddings could be used to achieve high precision and high recall. While some embeddings performed significantly better than others, all of them outperformed the frequent itemset baseline technique.

In the manual analysis of the results for the DBpedia experiments, we could gain several insights on the performance and the problems of the embeddings. For some potential synonyms, making a decision for or against being synonym was extremely difficult without looking into the data or having background knowledge on the relation's domain. In such complex cases, knowledge graph embedding-based synonym detection techniques are far from achieving perfect results. We believe that most complex cases may not be detected by any automatic method when additional background information is not available. Here, we think a human-in-the-loop process could help to further advance synonym detection in knowledge graphs.

4.3 Mining Relation Definitions

In the previous section, we have seen an approach for synonym detection in knowledge graphs, which works on arbitrary knowledge graph embedding techniques. We have seen that this method indeed achieves high-quality results on Freebase, Wikidata, and DBpedia, outperforming a frequent itemset baseline. However, as outlined in the conclusion, our manual evaluation has shown that many false-positive results have been returned. Often results were somehow related to each other but not synonymous. As an example, knowledge graph embeddings identify `north` and `south` as synonymous relations. This decision is incorrect. In this, but also in many other cases, it is difficult to understand why some synonyms are identified well while others are not. The latent nature of embedding techniques is not helpful to provide a better understanding.

In this section, we propose an interpretable technique building on logical rules to overcome this lack of explainability. Interpretability offers the possibility to use the method in a semi-automatic way to support manual cleaning efforts in a pre-processing step. Thus, a human supported by this synonym detection system could easily use his world knowledge to figure out that `north` and `south` are not synonymous.

The technique we have presented in an ESWC publication [58] uses Horn rule mining similar to the work on previous work on open knowledge graph canonicalization [34]. While Galárraga et al. mine equivalence rules of the form $r \Leftrightarrow r'$ directly from the knowledge graph and therefore require synonymous relations to have high overlap in subject and object entities, we have developed an indirect rule mining approach. Instead, we create logical definitions of relations, which even are understood by humans. Then definitions of relations are then matched against each other to find synonymous relations.

Example. For the relation `grandfather` a definition might look as follows:

$$father(x, y) \land father(y, z) \Leftrightarrow grandfather(x, z)$$

While for the relation `granddad`, we may find also a definition:

$$\texttt{father}(x,y) \land \texttt{father}(y,z) \Leftrightarrow \texttt{granddad}(x,z)$$

We observe that even though `grandfather` and `granddad` are different relations, they have identical definitions. Thus, we conclude that they are synonyms.

This section presents a rule mining technique, which identifies synonymous relations by matching their definitions with high quality. Furthermore, the technique offers good explainability in contrast to the embedding-based techniques we have presented before. In our evaluation section, we evaluate our technique on Wikidata and DBpedia against a frequent itemset baseline and the top embedding approaches from the previous section. We show that, indeed, rule mining is superior since it offers better results while still being explainable.

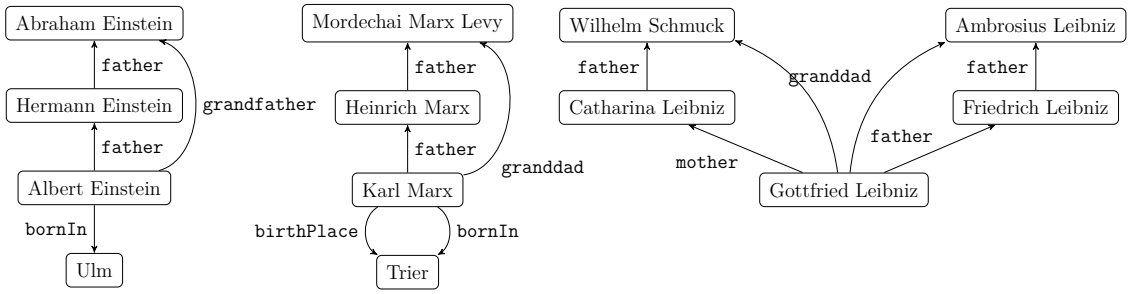

Figure 4.6: An example knowledge graph about persons and their ancestors. Nodes are entities and edges are relations.

4.3.1 Preliminaries

In this section, we shortly introduce Horn rules and metrics for Horn rule mining. Similar to before, we work with RDF-based knowledge graphs consisting of a set of subject, relation, object triples to build a method that identifies synonymous relations. Two relations r and r', having two distinct IRIs ($r \neq r'$), are called *synonymous*, when they represent the same real-world relation. Our notation of Horn rules over knowledge graphs stems from Galárraga et al. [37].

Each triple (s, r, o) can also be written in the form of a logical *atom* as $r(s, o)$ In an atom, not only entities (subjects or objects) are allowed, but also variables $x, y, z_1, z_2, \ldots$ from some universe of variables V. A rule is a logical implication consisting of a *body* and a *head*. The body is a conjunction of multiple atoms b_i ($i \in \{1, ..., k\}$) while the head is only a single atom: $b_1 \wedge ... \wedge b_k \Rightarrow r(s, o)$. If we have multiple positive and conjugated body atoms and a single positive head atom, we call the rule a *Horn rule*. If every variable in a rule occurs at least two times, the rule called *closed*. Throughout this work, we only use closed Horn rules.

Example. The definitions for `grandfather` and `granddad` we presented in the introduction were inspired by Horn rules. An underlying Horn rule is for example the following:

$$\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z)$$

The meaning of this rule is as follows. Entities matched to the body of the rule (subjects and objects are assigned to the variables) also match the head. If an entity is matched against y, this entity is supposed to be the father of the entity assigned to x. The entity assigned to z is supposed to be the father of the entity assigned to y. Then it is implied that z is the granddad of x. Usually, such (closed) rules are used for knowledge graph completion, i.e., to predict new facts. For the example knowledge graph depicted in Figure 4.6, we could match the rule as follows: `father`(Karl M., Heinreich M.) and `father`(Heinrich M., Mordechai M.L.) could be matched to the body, such that we predict the fact `granddad`(Karl M., Mordechai M.L.) using our rule. Hence, we predicted the already existing fact that Mordechai Marx Levy was the granddad of Karl Marx. Similarly, we could also predict the new fact `granddad`(Albert E., Abraham E.)

Such Horn rules are usually created in a mining process on an incomplete real-world knowledge graph. Thus they are also called statistical rules. Galárraga et al. present measures for assessing the quality of rules similar to classical quality measures for association rule mining [37]. The first measure is used to quantify the popularity of a rule, i.e., the number of instantiations of the rule in the knowledge graph. This measure is called *support*:

$$supp(B \Rightarrow r(x,y)) = \#(x,y) : \exists z_1, \ldots, z_n : B \wedge r(x,y),$$

with $z_1, \ldots, z_n$ being variables only occurring in the body B distinct from x and y. For our example rule from before, the support is 2, because the rule is supported by the instances Karl Marx, Gottfried Leibniz, and their respective ancestors. The rule $\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{grandfather}(x,z)$ has a support of 1. This classical support measure is an absolute measure for the frequency of instantiations of a rule in some knowledge graph. Thus the support is highly dependent on the size of the knowledge graph and the frequency of single relations in a knowledge graph.

To overcome this disadvantage, a relative support value, the *head coverage* was introduced [37]. It measures the absolute support relative to the occurrences of the head relation of a rule.

$$hc(B \Rightarrow r(x,y)) = \frac{supp(B \Rightarrow r(x,y))}{\#(x',y') : r(x',y')}$$

For our example rule, the respective head coverage is 0.66. The absolute support value was 2, while the head relation `granddad` occurs exactly three times.

An additional measure that is usually used for association rule mining is the *standard confidence*. The standard confidence assesses the prediction quality of a rule by comparing the number of true predictions in the knowledge graph relative to the number of all possible predictions that are made by the rule:

$$conf(B \Rightarrow r(x,y)) = \frac{supp(B \Rightarrow r(x,y))}{\#(x,y) : \exists z_1, \ldots, z_n : B}$$

The higher the confidence, the more a rule is justified by the knowledge graph. Regarding our example rule again, its confidence with regard to our knowledge graph in Figure 4.6 is 0.66. The body matches exactly three times to the knowledge graph, while the support is two. It implies that the rule predicts three facts, two of them already existing in the knowledge graph. The new fact that is predicted by the rule is Abraham Einstein is the granddad of Albert Einstein.

4.3.2 Mining Relation Definitions for Synonym Detection

In this section, we use existing rule mining techniques to find equivalence rules, which then are used for identifying synonymous relations. An *equivalence rule* or *synonym rule* states that two relations r_1 and r_2 are synonymous:

$$r_1(x,y) \Leftrightarrow r_2(x,y),$$

A synonym detection algorithm using rule mining to find synonyms tries to find two rules $r_1(x, y) \Rightarrow r_2(x, y)$ and $r_2(x, y) \Rightarrow r_1(x, y)$ which culminates to synonymity of r_1 and r_2 [34]. However, finding such a rule requires r_1 and r_2 to overlap in their extension, i.e., to share the same subject-object pairs in a knowledge graph.

Example. In our example knowledge graph depicted in Figure 4.6, we find the following two rules:$\texttt{bornIn}(x,y) \Rightarrow \texttt{birthPlace}(x,y)$ and $\texttt{birthPlace}(x,y) \Rightarrow \texttt{bornIn}(x,y)$. The first has standard confidence of 0.5, while the latter has perfect confidence of 1.0. Thus, we could assume them to be correct and infer the associated synonym rule $\texttt{bornIn}(x,y) \Leftrightarrow \texttt{birthPlace}(x,y)$.

But, since in real-world knowledge graphs, synonyms often stem from several sources, they may not be found by this rule mining idea since they do not share the same subject-object pairs within the same knowledge graph. Even though this simply rule-mining-based technique finds some synonymous relations, in real-world scenarios, a large part of synonyms cannot be found at all.

Example. The relations `granddad` and `grandfather` are synonymous in our example knowledge graph. They both describe the relationship between a grandchild and its grandfather but have different IRIs. However, the rule $\texttt{grandfather}(x,y) \Rightarrow \texttt{granddad}(x,y)$ has a support of 0 and thus cannot be found by any mining technique. Finding the respective synonym rule for the two relations is therefore not possible with this simple technique.

4.3.2.1 Mining Relation Definitions

To overcome the problem described in the previous example, we use an indirect mining approach to find synonym rules. We first mine so-called *relation definitions*. A definition is seen as a logical paraphrase of a relation through other relations.

Example. For the relation `granddad`, we may for example find the following definition:

$$\begin{aligned}\texttt{granddad}(x,z) \Leftrightarrow &(\texttt{father}(x,y) \wedge \texttt{father}(y,z)) \vee \\ &(\texttt{mother}(x,y) \wedge \texttt{father}(y,z))\end{aligned}$$

This definition says that a granddad is equivalent to being the father of some person's father or being the father of some person's mother. If we could find a similar definition for the relation `grandfather`, we could infer that they are synonyms.

Formally, we try to find a synonym rule $r_1(x, y) \Leftrightarrow r_2(x, y)$ indirectly by mining their definitions D.

$$r_1(x, y) \Leftrightarrow D \Leftrightarrow r_2(x, y),$$

From this rule, we infer the synonym rule by transitivity. However, state-of-the-art rule mining systems are build to find Horn rules. In contrast to definitions, a Horn rule only consists of a simple implication between body atoms and the single head atom. Mining in the opposite direction is not simply possible.

Example. Applying a rule mining approach to our example knowledge graph for the head relation `granddad` would give us two rules: (a) the paternal granddad

$$\texttt{father}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z)$$

but also (b) the maternal granddad

$$\texttt{mother}(x,y) \wedge \texttt{father}(y,z) \Rightarrow \texttt{granddad}(x,z).$$

Rule (a) has a standard confidence of 0.66 and a head coverage of 0.66, while the second rule has a standard confidence of 1.0 and a head coverage of 0.33. Both rules together describe what being a granddad means pretty well. The combination of both corresponds to the definition from before:

$$\begin{aligned}\texttt{granddad}(x,z) \Leftarrow &(\texttt{father}(x,y) \wedge \texttt{father}(y,z)) \vee \\ &(\texttt{mother}(x,y) \wedge \texttt{father}(y,z))\end{aligned}$$

This combined rule has a head coverage of 1.0 and a confidence 0.75. We see that the combined head coverage is the sum of the head coverages from both original rules. It is only true since the bodies match different entities. More generally, however, instantiations of different Horn clauses in a definition might overlap, which needs to be considered for head coverage computation of the disjunction of bodies by counting distinct instances only.

However, this rule still only expresses an implication and not the required equivalence between definition and head relation to achieving our goal of mining synonym rules. On the other hand, we have also observed that the disjunction of the bodies of the two rules leads to higher overall head coverage in the combined rule. A high head coverage implies that it is more likely that we observe the body when the head was matched. A head coverage of 1.0 means that whenever the head relation is matched to the knowledge graph, we also match the body. From this observation, we see that head coverage and standard confidence have something in common: Looking at their definitions again, we notice that both have the rule support in their numerator. For standard confidence, the denominator is the number of matches of the rule body. In contrast, for the head coverage, the denominator is the number of matches of the head relation. This implies the following equality:

$$conf(B \Rightarrow r(x,y)) = hc(B \Leftarrow r(x,y)).$$

A high head coverage and high standard confidence in a rule imply that the body and head are equivalent.

Using these observations, we are now able to define the idea of *propert definitions* for relations $r \in R$ as a disjunction of Horn clauses, i. e., $D = b_1 \vee ... \vee b_k$, such that the rule $D \Leftrightarrow r$ holds. This rule is fully supported by a knowledge graph when standard confidence and head coverage are exactly 1.0. However, in these cases, the definition and the respective relation share all their entities. It is usually not helpful for finding synonym rules. In practice, such high-quality rules are hardly found. Horn rule mining usually leads to high confidence rules with low head coverage. To achieve a high head coverage, we need to combine different rule bodies into

definitions covering diverse entities from the knowledge graph, such that the overall head coverage is high. This way, we are able to find relation definitions with high quality to now mine synonym rules for synonym relation detection.

4.3.2.2 Mining Synonym Rules by Matching Definitions

The problem when we use relation definitions for finding synonyms in heterogeneous knowledge graphs is that usually, no relations are found that have identical definitions. Usually, the data is too incomplete and too heterogeneous.

Example. When we now look again at the definition for `granddad` from above and a newly mined definition for `grandfather`:

$$\texttt{grandfather}(x,z) \Leftarrow \texttt{father}(x,y) \wedge \texttt{father}(y,z)$$

This rule has a confidence of 0.33 and a head coverage of 1.0. Thus we conclude that the definition of `grandfather` is the father of a father. Thus, it is incomplete, which is a typical problem that occurs in incomplete knowledge graphs. Hence, the definition is different from the definition that we mined for the `granddad` relation before. They only share one part of the definition. We therefore need to relax our idea of matching definitions to partial matches, such that we perform the following matching.

$$\begin{aligned} &\texttt{granddad}(x,z) \\ \Leftrightarrow &\texttt{father}(x,y) \wedge \texttt{father}(y,z) \\ \Leftrightarrow &\texttt{grandfather}(x,z) \end{aligned}$$

Since for `granddad`, we now leave out a part of the definition, the overall head coverage for its definition is lower. It may be a problem for the overall definition's quality, which may also reflect the quality of the resulting synonym rule.

We first describe the overall process: (1) We first perform a rule mining process on a knowledge graph to obtain definitions for all relations. (2) In a second step, we perform a pairwise matching between the definitions of all relations

In this second step, we aim at maximizing the overlap of the definitions of two relations since a high overlap ensures a high-quality synonym rule and, therefore, high-quality synonymous relations. To compute this overlap, we choose the Jaccard coefficient between definitions as $\frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}$. Bodies from a definition are identical if they are structurally identical. The result of this computation is a value between 0 and 1 for each relation pair of the knowledge graph. This Jaccard coefficient can be seen as some kind of confidence value for the matching process. In our `granddad` and `grandfather` example above, the Jaccard coefficient is 0.5.

The result of this process is a ranked list of pairs of relations by their Jaccard coefficient. If no matching partner could be found, the confidence value is 0.0. The top-ranked results of this list have the highest confidence of being synonymous relations. Low-ranked results may not be synonymous. Identifying synonymous relations from the knowledge graph boils down to picking a confidence threshold and counting all relation pairs from the list with higher confidence values as synonyms.

4.3.3 Evaluation

The presented experiments are based on a previously published work, and therefore several passages are quoted verbatim from the paper [58].

In our experiments, we evaluate our rule-based technique against a frequent itemset-based technique [1] and our knowledge graph embedding approach on two large real-world knowledge graphs. Our implementation, a description on how to reproduce the experiments, and the datasets are all available through our Github repository[4].

For all experiments, we employ an existing tool for mining Horn rules: we use AMIE+ [35] with a minimum head coverage of 0.005, minimum confidence of 0.05, and a minimum initial support to mine closed and connected Horn rules on the datasets. If the rule mining algorithm did not output new rules for more than 10 hours, we preliminary stopped the mining process and used the rules mined so far.

Overall, two experiments using seven baseline approaches are performed: (1) To assess whether the quality of synonym detection methods is ready for cleaning real-world knowledge graphs, we perform a manual evaluation of the quality of the system on DBpedia. (2) In the other experiment, we want to analyze the recall and precision of synonym detection techniques on synthetically created synonyms in Wikidata.

Overall, we compare the approaches on two large real-world datasets Wikidata and DBpedia. Since both datasets have several hundred million triples, which is unfeasible for training knowledge graph embeddings as well as for mining rules in a feasible time, we stick to the sampled datasets that have been built in [55]. It also allows for a better comparison of our results to previous works. In their work, the authors have presented a sampling technique that keeps triples with every existing relation in the respective knowledge graph while reducing the overall number of triples. Our gold standard datasets containing our manually labeled synonyms for DBpedia and the synthetic synonyms for Wikidata are available online[5].

Frequent Itemset Baseline. The approach presented in [1] uses frequent itemset mining to detect synonymously used relations to perform query expansion. In this work, we used the implementation and results of this baseline from [55]. In that work, we re-implemented the approach using *Range Content Filtering* and *Reversed Correlation Coefficient* as described in the original paper using Python and Spark. The implementation of the approach is also openly available on Github. As an input parameter for frequent itemset mining, the approach requires the user to provide a minimum support value. For both experiments, a grid search optimizing for optimal F1-measures was performed.

Knowledge Graph Embedding Baselines. In our previous work that was described in Section 4.2, we have shown that knowledge graph embeddings might be used to detect synonymous relations by using outlier detection techniques on the relation representation in state-of-the-art embeddings. In this section, we only take the top 6 embeddings with the metrics that worked best: TransH [117], TransD [50]

[4] `https://github.com/JanKalo/RuleAlign`

[5] `https://doi.org/10.6084/m9.figshare.11343785.v1`

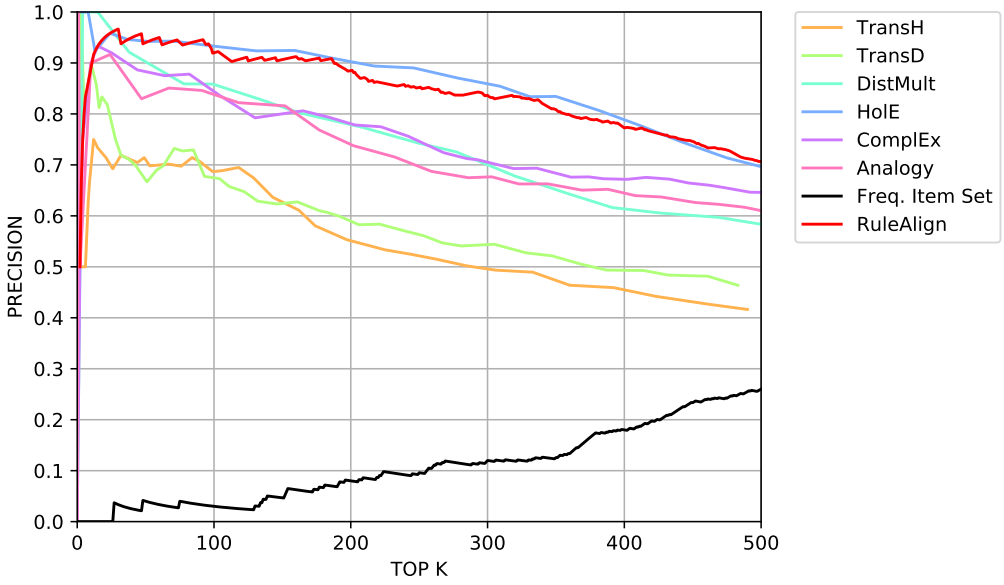

Figure 4.7: Experimental results from our approach RuleAlign in red to several baselines on DBpedia manually evaluated with precision@k up to $k = 500$.

ComplEx [110], DistMult [121], ANALOGY [67] and HolE [82]. All these techniques achieve high quality in the top results. The recall, however, is problematic in some of the presented experiments. We further analyze the differences of the fundamentally different approaches embeddings vs. logical rules in various settings here.

4.3.3.1 Manual Quality Evaluation in DBpedia

The DBpedia sample comprises 12 million triples with around 15,000 different relations with several natural synonyms, ranging from rare synonyms only occurring in around 100 triples up to synonyms being part of hundreds of thousands of triples. The evaluation on DBpedia is performed manually for the top 500 results of each of the approaches classifying pairs of relations in either being synonyms or not. For the baseline approaches, we rely on the datasets classified from Section 5.2 extended by a manual classification performed for our newly proposed approach.

In this experiment, we have performed a manual evaluation for the precision@k up to $k = 500$ on a DBpedia sample comparing eight different approaches. The results are presented as line graphs in Figure 4.7.

The frequent itemset-based baseline has an increasing precision for higher k values due to a ranking function that assumes that synonymous relations are not occurring for similar subject entities. This assumption is not true for DBpedia. The precision for this baseline always is below 30% and also does not exceed 30% for k values above 500. The best embedding-based baseline is HolE, having a maximum precision of over 90% in the top 200 results and precision around 70% at $k = 500$.

Our approach, presented as RuleAlign in red, shows the best results in this experiment together with the embedding model HolE, finding at least 352 correct synonyms. Overall, the number should go into the thousands when we extended our manual evaluation. In comparison to a direct rule mining approach for equivalence

Table 4.1: Matched relation definitions mined from DBpedia as an explanation for the result.

Relation	Definition
`grandsire`(x,z)	`sire`(x,y) ∧ `sire`(y,z)
`nationality`(x,y)	`stateOfOrigin`(x,y)
`nationality`(x,z)	`birthPlace`(x,y) ∧ `country`(y,z)
`north`(x,y)	`east`(y,z) ∧ `northeast`(x,z)

rules, our indirect approach finds at least 77 correct synonym pairs on our DBpedia dataset, which cannot be found by the other approach because they have no support.

As an additional feature, our approach is able to propose explanations for the synonym predictions in the form of relation definitions. The top explanations are having a high head coverage, covering lots of entities, and have high confidence. In Table 4.1, we present some example definitions from DBpedia. Since, for many relations, around 100 Horn clauses are in the definition, we only present top matched Horn clauses. These explanations are natural definitions of the respective relations that would also be used in the real world. Note that besides these human-readable example definitions, many synonym pairs are entirely different in their respective IRI labels, e.g., "dbp:ff" (father of the father) and "dbp:grandsire" and are therefore difficult to be identified by humans without our automatic data-driven approach.

A closer look at our predictions reveals some shortcomings of our approach. First of all, our approach is not able to distinguish the gender within some relations. We classify, for example, `father` and `mother` as synonyms because no rule captures the gender correctly. One reason for that is that gender is only mentioned as a literal, ignored by the rule mining approach. A second problem is relations that hardly are distinguished by their data instances because they are extremely similar. As an example `firstDriver` and `secondDriver` representing a person's placement in a race, cannot be distinguished. Furthermore, false-positives in the form of hyponyms as for example `genre` and `musicGenre` are returned.

4.3.3.2 Precision-Recall Evaluation in Wikidata

The Wikidata sample has more than 11 million triples and more than 1,500 relations. In contrast to DBpedia, it is supposed to be free of synonyms due to intensive manual curation. Therefore, we have introduced synthetic synonyms here by randomly re-naming existing relations similar to our evaluation in Section 4.2. For the triple (Albert E., father, Hermann E.), we instead use (Albert E., father_synonym, Hermann E.). Thus, the relations `father` and `father_synonym` are treated as synonyms but never co-occur for the same subject-object pair. Overall, 343 synonymous relations have been introduced that need to be identified for the approach. A more detailed description of the creation of the dataset can be found in the previous section.

The second experiment measures precision and recall for the Wikidata sample. Our results regarding this experiment are presented as precision-recall curves in Figure 4.8. We again start by having a look at the frequent itemset baseline in black.

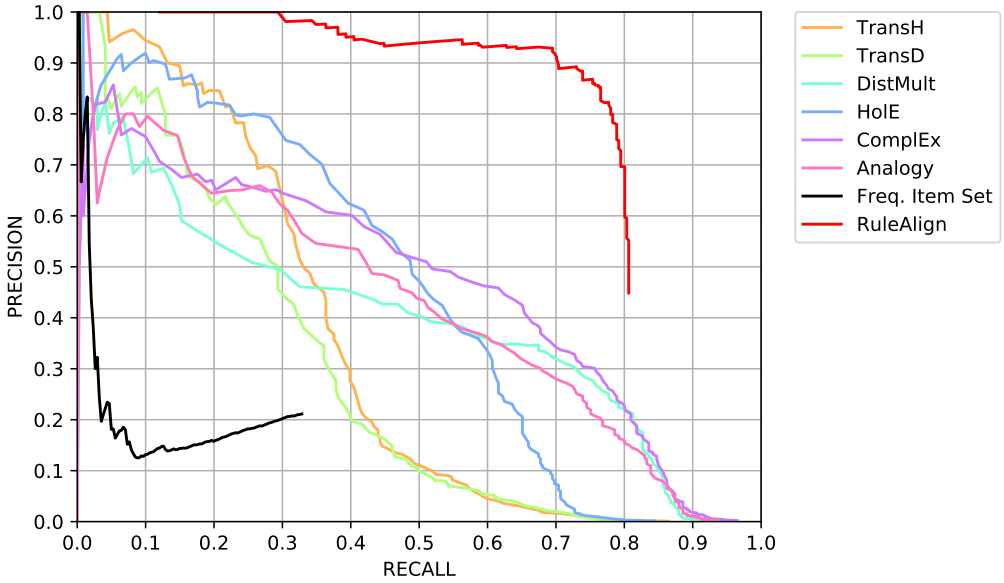

Figure 4.8: Experimental results from our approach RuleAlign on Wikidata. We provide a precision-recall analysis for synthetic synonyms.

It starts with high precision for low recall values and then drops sharply to under 20%. The maximum precision is at 21% at a recall value of around 35%. Due to the minimum support value that leads to the best F1-measure, no higher recall value is achieved here. Embedding-based approaches achieve a high precision up to a recall of 30%. The best approach is again HolE, starting at 90% precision for a recall of 10% and precision of 10% for 70% recall. In contrast, our approach (red) is having a perfect precision for recall values up to 30% and still a precision over 90% for a recall of 70%. The recall of our approach sharply drops, only achieving around 80% recall.

For Wikidata, our approach achieves extremely high precision but also has problems in the recall due to two reasons: (1) For 32 relations, no rule could be mined due to the minimum head coverage in the rule mining process. (2) The other synonyms could not be found since none of the mined rules fulfilled our minimum confidence threshold. The few false positives that have been returned by our approach often were hyponyms instead of synonyms.

The rule-based approach matching data-driven relation definitions for detecting synonymous relations achieves high precision. In both datasets, we could observe that a high Jaccard coefficient often implies that the respective relation pair is synonymous. In the Wikidata experiment, all pairs with confidence above 0.9 are synonyms, and also, in DBpedia, a high Jaccard coefficient leads to good results.

However, in DBpedia, only a few synonyms with high confidence could be found. For lower Jaccard coefficients, a higher proportion of false positives is returned because these relations often were in a hyponym relation. This problem could be solved by an improved matching process that also takes into account the head coverages of the rules when computing the Jaccard coefficient. However, this might further decrease the recall of our approach, which has already been observed as a

problem for the Wikidata dataset. The simple Jaccard coefficient, as used in this work, achieves a high precision with a reasonable recall.

A low recall could also be prevented by mining rules with lower head coverage, by mining more expressive rules, or by decreasing the minimum confidence threshold. In turn, this might further decrease the performance of the rule mining tool, resulting in enormous rule sets.

Several false positives that were returned in DBpedia had a high overlap in their data instances and, therefore, also similar definitions. These relations were similar, but from the labels or IRIs, we observed that they were not a synonym. These cases are hardly identified in a data-driven fashion because they often need detailed domain knowledge.

4.3.4 Discussion

In this section, we presented a novel approach for detecting synonymous relations in real-world knowledge graphs in a purely data-driven way using Horn rule mining. One key advantage of this technique is that it is fully explainable and, therefore, is easily integrated into a human-in-the-loop process for cleaning heterogeneous knowledge graphs. We were able to evaluate the performance of our technique on two large knowledge graphs against embedding methods that were also developed by us. The precision achieved by the rule mining approach was even higher than for the knowledge graph embedding approaches.

However, our manual analysis of the results also gave us the impression that this rule-based approach is close to the maximum precision that a purely data-driven synonym detection for relations can achieve. Many false positives detected by our technique had an extremely high overlap in their subject and object entities. Only the IRI revealed that the relations have different semantics. Such cases can only be solved when additional knowledge is integrated into the matching process. It can either be background knowledge in the form of textual descriptions of the relations or some kind of domain knowledge.

4.4 Conclusion

In this chapter, we have discussed the problem of relation heterogeneity. Instead of classical matching problems between exactly two knowledge graphs, detecting synonymous relations within a single knowledge graph brings various new problems that existing techniques cannot deal with. At the point of publication of our ideas, only a few ideas on dealing with synonymous relations have been published [1]. Some novel ideas similar to our work have been published lately [23, 124]. Also, some work on hypernyms and hyponyms in single knowledge graph environments was recently published [51]. However, the existing body of work is scarce, and the problem is far from being solved.

This chapter presents several novel methods for detecting synonymous relations using knowledge graph embeddings and Horn rule mining. Both methods achieve very high precision for synonymous relation detection in large real-world knowledge graphs and outperform an existing method by far. We showed that while knowledge graph

embeddings show promising results on large-scale heterogeneous knowledge graphs on synthetic synonyms and actual synonyms in DBpedia, they lack explainability. In follow-up work, we could show that an indirect rule mining approach based on Horn rule mining finding relation definitions outperforms the novel embedding techniques while guaranteeing good explainability. We have shown that in a real-world scenario on DBpedia, several hundred synonymous relations could be identified with a precision of over 70%. Even though these results are not perfect, we believe that the rule mining idea offers the possibility to support human workers in cleaning knowledge graphs.

For the future, it would be interesting to combine both works with our work from Chapter 3 so that negative transitivity effects are minimized when more than two synonymous relations are found in a knowledge graph. First experiments have shown that the transitivity problem is more difficult in a single-knowledge graph environment since usually only very small equivalence classes are created. Furthermore, we would like to investigate existing methods for hypernym, hyponym, and inverse relation detection. They are, similar to synonymous relations, a heterogeneity issue that might cause incomplete query results. We think that our techniques could, with some changes, be adapted to also these issues. We also plan to extend the rule mining approach to use more expressive rules so that the precision and recall can be improved even further. However, this is quite difficult due to the major performance issues that come with more complex rules.

5

Avoiding Heterogeneity by Implicit Knowledge Representation

In previous chapters, we have discussed several heterogeneity issues that come with large real-world knowledge graphs built from various sources. We have seen that research in the field of entity matching, ontology matching, and relation matching has a long history of automatically integrating knowledge. In recent years several advances in machine learning, including novel embedding-based techniques or large artificial neural networks for performing supervised classification problems, have been made. Hence, methods for automatic data integration have been improved. However, we are still far from being able to solve all heterogeneity problems. Novel techniques either require massive amounts of training data or are only evaluated on artificial benchmarks that are not representative of real-world problems. As shown in our chapter on entity heterogeneity, state-of-the-art entity matching techniques cannot work in realistic multi-knowledge graph scenarios. For relation matching, only a few methods exist at all. As discussed in Chapter 3, synonymous relations still need to be resolved by humans since automatic techniques cannot achieve sufficiently high precision.

Recently, a novel paradigm that could be used to overcome most of these heterogeneity issues was introduced. Instead of explicitly storing knowledge in a knowledge graph, we could query large neural language models that implicitly contain knowledge from millions of unstructured texts [90]. Neural language models learn to predict the next word/sentence given a sequence of words.

Example. A query asking for the `birthplace` of Albert Einstein could be translated into a natural language sentence:
Albert Einstein was born in [MASK].
The [MASK] is completed by the language model with the most probable word that fits that position based on previously read texts. It returns a list of words together with their probabilities. Among these results is the word `Ulm`, the city of Einstein's birth, which then is returned as a result.

Petroni et al. have shown that language models indeed encode large amounts of knowledge and could be used instead of a knowledge graph [90]. Some heterogeneity issues become obsolete since the language model internally solves several heterogeneity issues that are present in text. Instead of storing explicit facts in a heterogeneous knowledge graph that has to be cleaned and integrated correctly, a language model offers the possibility to implicitly store knowledge from unstructured data.

This chapter further investigates the idea of using a language model as an implicit knowledge graph by building a hybrid querying system. Instead of entirely relying on the language model, we instead build a system that answers queries from a knowledge graph and language model in a joined fashion. This way, we can query a knowledge graph, even if its heterogeneity leads to incomplete query results while complementing these results with additional knowledge from the language model. Hence, a heterogeneous knowledge graph profits from the knowledge in the language model. On the other side, also the results of the language model are further filtered using the explicit semantic information in the knowledge graph. In a hybrid system, both techniques profit from each other.

Research Questions.

- *How do we use implicit knowledge representation to support querying in knowledge graphs?*
- *How do we combine the advantages of knowledge graphs and neural language models?*
- *Which kind of knowledge is implicitly stored in neural language models?*

Contribution. This chapter explores the idea of using neural language models as an additional source of knowledge. We present the hybrid approach KnowlyBERT as a combination of the language model BERT and the knowledge graph Wikidata to perform query answering. The main contributions of this chapter are based on the work published in [56]. In several large-scale experiments with around 6500 queries, we show that KnowlyBERT outperforms two state-of-the-art baselines on the real-world knowledge graph Wikidata. The implementation and all our experimental data are openly available[1].

Outline. Section 5.1 gives an overview of the related work on querying language models, extracting knowledge from language models, and question answering on text. Some preliminary notions about language models and querying language models are presented in Section 5.2. In Section 5.3, we present the several steps that KnowlyBERT needs to perform high-quality query answering. The quality of KnowlyBERT is evaluated in Section 5.4. Here, we provide a detailed analysis of the advantages and disadvantages of a language-modeling-based querying system. In the last section of this chapter 5.5, we discuss our results and look into future research directions.

[1] `https://github.com/JanKalo/KnowlyBERT`

5.1 Related Work

In contrast to storing knowledge explicitly in knowledge graphs, recent works have shown that there are several techniques to store knowledge implicitly. On the one hand, a large body of work on using language models instead of knowledge graphs has been proposed. In this direction, language models were improved to better store entity knowledge. Some recent work uses a language model as a database interface to query heterogeneous data. In another direction, people heavily work on question answering from texts. A technique that could be used instead of standard knowledge graphs since it does not require mapping knowledge to fact triples.

5.1.1 Language Models as Knowledge Graphs

Recent work has shown that large neural language models encode large amounts of world knowledge and thus may be used to support a variety of knowledge-intensive tasks [16]. To the best of our knowledge, Petroni et al. have been the first who proposed to use language models as knowledge graphs by directly extracting facts from pre-trained language models [90]. They show that standard language models may be queried similar to large knowledge graphs and achieve high quality and good recall on several datasets. Concretely, the authors have shown that simple basic graph pattern queries may be encoded into natural language sentences, such that the completed words of the sentence serve as answers to the original query. For each possible relation of a query, the authors have manually created so-called *template sentences* that serve as a query for the language model. These templates are then used to complete a sentence and predict a word to complete a triple, as demonstrated in the introduction. This method only predicts entity labels consisting of a single word, excluding almost all persons from being a query answer. Furthermore, it only provides words as answers, but not knowledge graph IRIs. An entity linking step is not performed yet. The evaluation is performed on several relations from Wikidata, but also on a variety of other datasets. They achieve high accuracy of 32% on the T-Rex datasets [29], comprising 41 different Wikidata relations. We evaluate our system on the same relations from T-Rex, but a larger dataset also comprising multi-word entities.

In an extension of Petroni's work, it was shown that the quality of the language model's predictions is highly dependent on the used template sentences. Manually choosing the wrong templates may result in low answer correctness. To overcome this problem, Bouraoui et al. propose to automatically generate sentence templates [14]. They show that indeed they double the accuracy of Petroni et al. This approach is the basis for our template generation process for KnowlyBERT.

A similar approach was presented by Jiang et al. [51]. The authors have created a slightly different technique to create templates for querying language models but show comparable results.

Very recently, a pre-print about language model-based databases, neural databases, was published [108]. A neural database is using textual data for storing knowledge,

updating knowledge, and natural language questions instead of structured queries. Its promise is that querying schema-free data becomes available, guaranteeing a high query answering quality. In general, the idea proposed by Thorne et al. is comparable to what we propose in this chapter and to what Petroni et al. have proposed [90] because all works are about using querying to improve querying. Among others, two key points of the neural database idea are: (1) Instead of querying the pre-trained language model, the neural database consists of a set of facts in natural language sentences (the database) with the language model as a query processor and (2) neural databases allow for several query types, ranging from simple fact queries over join queries to aggregation queries.

Neural databases are built by fine-tuning a large pre-trained language model for query answering on a set of textual facts (the database) using a training set with a large number of different query types and their respective answers. This process is comparable to what is known from standard (closed book) question answering using language models. Since a language model's input is usually restricted to a couple of hundred/thousand tokens, the basic idea is not scalable for larger databases. For question answering, usually, an information retrieval component is used to identify relevant facts for answering a certain query before using the actual language model. However, this approach would not allow for aggregation queries over larger databases. Hence, Thorne et al. propose an idea on how to parallelize the process using multiple language models.

In the experiments, a database with more than 8,400 facts is evaluated on more than 14,000 queries. The authors show that simple fact and join queries are answered with a precision of around 80% and aggregation queries around 60%. The recall of this approach is usually above 90%.

Neural databases, in general, provide an interesting idea for overcoming heterogeneity issues in knowledge graphs and databases in general because a high precision/recall is achieved for diverse query types answered directly from text. The basic idea is significantly different from ours since they rely on fine-tuning a language model, while we combine knowledge graph knowledge only with a pre-trained language model. However, we believe that this work offers great potential for future research in this direction.

Knowledge-enabled Language Models. Language models themselves are only trained on words, leaving out the valuable entity/relation knowledge, which is contained in large-scale knowledge graphs. The combination of these two sources of knowledge can, for example, be done by injecting knowledge graph knowledge directly into the language model during training to improve knowledge-intensive tasks, such as question answering, relation classification, or entity typing. This knowledge injection is done by performing entity linking on the text corpus before pre-training a language model [122] or by directly incorporating knowledge graph triples into the training of a language model [91, 116, 120].

Poerner et al. have shown that using a standard language model as a knowledge graph has some problems [91]. In this work, the authors modify the language model by including external entity knowledge. The authors align the embeddings from the BERT language model with pre-trained entity vectors from the Wikipedia2Vec

corpus to give the language model a notion of entities instead of just words. The big advantage of this technique in contrast to other entity-aware language models is that a new pre-training of the language model is not required. The entity knowledge is injected into an existing model. The new model E-BERT is then evaluated on an improved LAMA dataset and compared to the work of Petroni et al. from [90]. It is shown that E-BERT is not only outperforming Petroni et al. with a precision@1 of 56.1% but also several other baselines that use language models for knowledge-intensive tasks. Later the model is evaluated on two applications: entity linking and relation classification. For relation classification, the other baselines are slightly outperformed with a macro-averaged F1 measure of 88.38%. In contrast, entity linking is working slightly worse. E-BERT could be used to further improve our system KnowlyBERT, since the integration of knowledge into the language model seems to be very valuable, increasing the quality in query answering tasks significantly.

Another work in this direction circumvents the restriction to single-word entities by defining a new fine-tuning task on masked language models [120]. They directly include entity knowledge from the knowledge graph to train masked language models. However, in their work, they restrict to a small amount of most popular entities from Wikidata only. Covering a large number of entities, including rare entities, would dramatically increase the computational effort for training. Thus, this approach is not suitable for general query answering in knowledge graphs.

5.1.2 Open Domain Question Answering

Answering factual queries about entities may also be regarded from the perspective of NLP research. Natural language question answering is a well-researched topic. In general, two fields are distinguished, answering questions where the answer is found within a small text paragraph, called closed book question answering, or open-book question answering (or open domain question answering) where a possible answer is found in some larger text corpus [19]. The general idea of open domain question answering is as follows: An information retrieval component is used to search a large corpus of text to retrieve relevant text passages that contain the answer to the question. This retrieval component often involves standard information retrieval ranking techniques, such as TF-IDF or BM25. Then a standard question answering method (the reading component) is used to mark the word span within this text passage that is the answer to the question.

Novel techniques for open-domain question answering have shown that using a dense passage retrieval mechanism for retrieving relevant passages together with a standard reading component outperforms methods using standard information retrieval techniques [59]. The retrieval model is trained on a small set of questions and relevant text passages. It then is able to create dense vector representations of questions and arbitrary text passages. The similarity between a question and a passage is used to determine whether it is relevant or not. Relevant passages for some questions are then put into a standard reading component for question answering that identifies an answer span within this passage that is used as an answer to the question. In the experiments, the dense passage retrieval approach is evaluated on

several standard benchmark datasets. The reported accuracy varies between 20% and 60%, thus far from being perfect.

Knowledge graph query answering, as presented in our work, resembles question answering, since in both tasks, some query, either in natural language or in a structured query language, needs to be answered. In contrast, however, the answer in knowledge graphs are entities with a unique identifier, whereas natural language answers may be ambiguous entity names.

5.2 Preliminaries

Our hybrid query answering system is built to work with RDF-based knowledge graphs and masked language models to answer SPARQL queries. In this preliminary section, we introduce the basic ideas of language models and how to query language models with SPARQL.

An RDF-based knowledge graph was defined as a set of triples $KG \subseteq E \times R \times (E \cup L)$ in Section 2.1. Querying such a graph is usually performed by the query language SPARQL. We restrict ourselves to simple, entity-centric queries with only a single BGP, where the variable may either be in the subject position or the object position. We call queries with a variable in subject position a *subject query* and a query with a variable in object position an *object query*.

Language models are statistical models trained on a large text corpus to learn to predict upcoming words given a sequence of words. Large transformer-based (a new architecture for artificial neural networks) language models have recently gotten much attention in natural language processing. In this work, we focus on the masked language model BERT [25]. During BERT training, the model is fed with sentences from a large text corpus to either learn to predict a word within the sentence or the next sentence given the previous sentence. This process is called self-supervised training since no additionally labeled training data is needed. After an extensive and resource-intensive pre-training, which is performed on large amounts of text for several epochs, the language model predicts words in arbitrary sentences, given the knowledge it has gathered in the training process. As an example, the sentence *Albert Einstein was born in ...*, would be completed by the most probable word that fits the context of the sentence. Usually, the language model forms grammatically correct sentences. From the name Albert Einstein, it might infer that this is a German-sounding name that fits a German city. The language model might have even stored that some text mentioned that Einstein was born in Ulm. So it may correctly predict the word `Ulm` which is the correct birthplace of Einstein. Following the idea of Petroni et al. [90], this mechanism is employed to answer simple BGP SPARQL queries.

Example. To illustrate the querying process, we use our running example, asking for the birthplace of Einstein again.

```
 SELECT ?birthplace
 WHERE (
<Albert_Einstein> <bornIn> ?birthplace.
 )
```

This SPARQL query may be transformed into the sentence *Albert Einstein was born in [MASK].* It is used as an input for the language model. The language model then returns a list of the most probable words and their confidence values.

For each relation in the knowledge graph, we need sentences that are used to translate queries. We use a natural sentence with two placeholders: one to be substituted with the subject entity, the other one with the object entity of some relation: *[S] was born in [O].* This natural language query is called *template*, sometimes also known as prompt.

Previous work has shown that the querying process is improved by concatenating additional *context* sentences to the template query [89]. If relevant context sentences about the entity of interest are picked, the precision may be improved by around 30%. We have implemented this idea of Petroni et al. by using the beginning of Wikipedia abstracts for the respective entities as a context paragraph.

Example. Hence, if our query is about Einstein's birthplace, we retrieve the first five sentences of Einstein's Wikipedia article and append it to the template sentence separated by a separator token ([SEP]). We provide context with the first sentence of Einstein's Wikipedia article as follows:

```
Albert Einstein was born in the city of [MASK].[SEP]
Albert Einstein was a German-born theoretical physicist who
developed the theory of relativity, one of the two pillars
of modern physics (alongside quantum mechanics).
```

To answer the query which was translated to a template, the language model now returns a list of words that replace the [MASK] token from the sentence. The result list returns single words together with a confidence value.

1. *a - 0.95*
2. *the - 0.93*
3. *Germany - 0.86*
4. *Ulm - 0.79*
5. *Berlin - 0.70*

This list usually contains several words which are grammatically correct but are not necessarily a valid answer to the query. In the case of our example, we also have the correct answer *Ulm* in the set of results, but also several results that need to be filtered out.

Thus, to build a functioning querying system, we further filter the result list of the language model by applying multiple cleaning steps which make use of the semantic information in the knowledge graph.

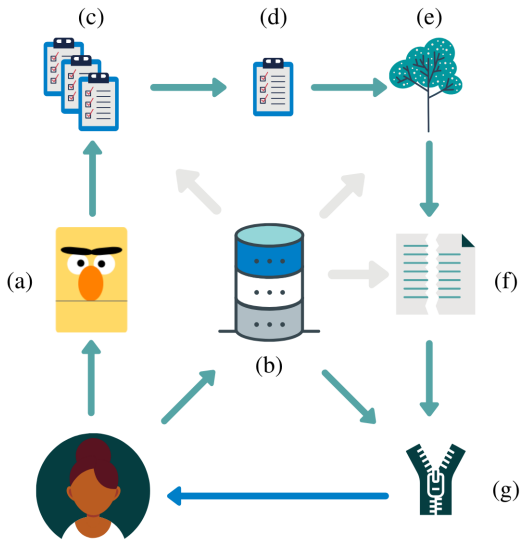

Figure 5.1: An overview of the query answering system KnowlyBERT.

5.3 Query Answering with KnowlyBERT

This section gives an overview of our hybrid query answering system *KnowlyBERT*, which combines the neural language model BERT with a knowledge graph to perform query answering. First, we provide an overview on KnowlyBERT while we go into the various components presented in the overview in the following sections.

5.3.1 System Overview

Let us first provide a short overview of our KnowlyBERT query answering system as depicted in Figure 5.1. The input for KnowlyBERT is a SPARQL query. In a first step, this query is transformed into multiple natural language sentences (templates), sent to the language model BERT (a). Simultaneously, the original SPARQL query is sent to the knowledge graph (b). The language model returns a list of results for each query sentence (c). These lists are integrated into a single list (d) and filtered by our semantic type filter (e). Before returning the results, the language model's top results are selected by a thresholding procedure (f). Finally, results from the knowledge graph and the language model are combined, duplicates are removed, and returned to the user (g).

5.3.2 Template Generation

As described in the previous section, language models are built to complete natural language sentences based on the texts it was trained on. To integrate this behavior into a querying system, we need a method that transforms an input SPARQL query into a template.

This transformation process is either performed in a manual fashion by predefined sentences similar to Petroni et al. [90] or by automatic systems, as shown in [14, 52]. Since previous works have shown that automatically generated templates outperform

manually designed templates significantly, for KnowlyBERT, we adapt the ideas from one of the automatic template generation systems from Bouraoui et al. with some improvements [14].

Example. We first provide a short example to explain how templates for the relation `birthplace` are mined. Instances of the relation are queried from the knowledge graph:

- (Albert Einstein, Ulm)
- (Max Planck, Kiel)
- (Thomas Mann, Lübeck)

From a text corpus, sentences containing any of these pairs are extracted. For example, a sentence about Einstein might be *The famous scientist Albert Einstein was born in Ulm.* The resulting template for this sentence is *The famous scientist [S] was born in [O].* Next, we use the template to predict the birthplace of Albert Einstein: *The famous scientist Albert Einstein was born in [MASK].* to predict possible objects.

Possible predictions for the sentence may contain popular cities of Germany, but maybe also the correct answer:

1. `Berlin`
2. `Frankfurt`
3. `Ulm`
4. `Kiel`

The size of the overlap between this list and all possible objects from the `birthplace` instances in the knowledge graph is computed. The overlapping instances are `Ulm` and `Kiel`, hence the overlap is 2. A similar process is performed analogously for the different object entities predicting possible subjects. The sizes of the overlaps are summed up to reflect the overall score of a template.

In detail, the template generation process can be described as follows. Sentences, including the entity pairs of the respective relation, are extracted from a large text corpus, which was tagged with entities beforehand. For some relation r, we extract all subject-object entity pairs, (s, o) from a knowledge graph, such that the subject s is in relation r with object entity o. We identify all sentences of the text corpus, such that a sentence contains exactly the two entities s and o, to heuristically identify the sentences expressing only the relation of interest r. To prevent sentences from being too specific, long sentences with more than 15 words are ignored. If more than the two entities of interest are tagged in the sentence, the sentence may express a more specific relation. Thus these sentences are also discarded. This idea is similar to what Bouraoui et al. have proposed in [14]. However, our text corpus has been pre-processed by a co-reference resolution method replacing pronouns by the respective entity names [29].

Additionally to existing methods, we also implement a basic duplicate detection technique among the identified sentences. This step aims to prevent similar templates for the same relation from being selected by our template generation method, such that the templates are as diverse as possible. We measure the pairwise similarity

between all possible templates of the same relation r by a basic sequence pattern matching method finding the longest sub-sequence between two sentences [97]. If a similarity exceeds 0.8, the sentences are put into the same cluster. To prevent long chains of sentences within the same cluster, we apply a community detection method (Girvan-Newman) on each cluster's similarity graph, possibly splitting up clusters into smaller clusters. This method is similar to what we have seen in Chapter 3 for identity graphs of entities. In the end, a single representative template is picked from each cluster.

In the final step of the template generation method, we create rankings of templates for each relation by giving each template a score. A template is scored by checking its predictive performance using the existing (s, o) pairs for the respective relation. For each template, we either predict subjects or objects. The overlap between the predictions and the set of all subjects/objects for the respective relation from the knowledge graph is computed. This overlap is computed for subject entities and analogously for object entities. In the end, the two overlap sizes for one sentence are summed up and used as a score. We save the top-ranked templates for each relation.

While this ranking method is problematic, since templates that never predict the correct results may be ranked high, practical evaluations have shown that the method shows promising results in ranking templates while still showing a manageable runtime.

5.3.3 Querying Language Models and Combining the Results

Existing techniques for querying language models are restricted to work with entities whose labels have exactly one word, excluding a large proportion of knowledge graph's entities. For example, most persons are not returned as an answer to any query since their labels consist of a first name and a last name. To overcome this problem, we introduce using multiple [MASK] tokens in a query template. Thus, instead of a single word, multiple words may be returned.

Example. Concerning our Einstein example, we change our query into the following query:

```
SELECT ?person WHERE{
?person <bornIn> <Ulm>}
```

One possible correct answer to this query is Albert Einstein, an entity label consisting of two tokens. Instead of just querying with the sentence "[MASK] *was born in the city of Ulm.*", we additionally use the two queries "[MASK] [MASK] *was born in the city of Ulm*" for returning two tokens and "[MASK] [MASK] [MASK] *was born in the city of Ulm*" if we want to find answer entities with three tokens. Hence, it is possible to return the correct answer: Albert Einstein.

One problem with this idea is that independent predictions for each [MASK] token are made. It leads to a single result list for each [MASK] token. Therefore, a query with two [MASK] tokens returns two independent result lists of words that

need to be integrated. Since combining every two words pairwisely would lead to many non-sense answers, we check whether a combination leads to a valid entity label, which is found in the knowledge graph using a fast trie data structure on the knowledge graph's entity labels. All other combinations are discarded. Valid combinations are assigned the average of the individual words' confidence values. The resulting list may contain entities of several length:

1. `Albert Einstein - 0.95`
2. `Einstein - 0.45`
3. `Germany - 0.22`

We have restricted the process to use at most three [MASK] tokens per query in this work. More [MASK] tokens can be used to increase the recall of the answers. However, it comes with increases in the query answering time.

Aggregating Results from Multiple Templates. Our method did not only use a single query template but five different templates for each relation to improving the recall of the querying system. As a result, we obtain five independent result lists that need to be integrated. Again, we stick to an idea similar to Bouraoui et al. [14]. The result lists are merged. If an answer occurs in more than one of these lists, we only use the largest confidence value of these answers and remove the remaining duplicates. A high predictive value for one template and a low predictive value for another template may indicate uncertainty. If the difference between the lowest and the highest confidence for the same answer exceeds the threshold of 0.6, the answer is discarded. This step has shown a significant improvement in precision in all our experiments. Higher thresholds further improve precision but lead to high losses in recall. The chosen threshold of 0.6 is a good compromise between precision and recall.

5.3.4 Semantic Type Filtering

When we look at the result list above, we notice that some result candidates have a type that does not match the query.

Example. `Germany` cannot be in the subject position of the relation `birthplace`. The subject may, among others, be a person, a movie character, or an animal, but not of type country. Such an incorrect answer is detected by using existing semantic knowledge from the knowledge graph and thus can be removed.

Since a relation may allow several different subject and object entities, we looked at the type distributions for subjects and objects. We separately build a frequency distribution of their classes and the direct superclasses of these classes for the subject and object entities. The most frequent classes are used as valid types for the relation's subjects or objects. If the language model returns results whose type does not fit these frequent classes, we remove it to guarantee high precision.

Entity Disambiguation. Since entity labels cannot be uniquely mapped to knowledge graph IRIs, a so-called entity disambiguation step needs to be performed before answering the query. `Albert Einstein` is the name of the famous scientist

and the name of several paintings (often portraits of Einstein), a high school, or a music album. The semantic type filter already does the most challenging part of the disambiguation step and removes all non-person entities. In some cases, after type filtering, we still are left with more than a single entity. Detailed disambiguation is not possible because we lack additional information on the result entities. Thus, we decided to exclude rare entities by a popularity filter. Entities that never occur in the object position of any triple in the knowledge graph are discarded. If multiple homonyms exist, the most popular entity is returned as an answer. We have also evaluated an additional filtering step using a knowledge graph embedding here. However, our evaluation has shown no benefits in precision without large losses in recall when we additionally employ embeddings.

5.3.5 Thresholding

After the previous step, we obtain a long result list still containing low confidence, possibly incorrect, answers at the bottom of the list. To only return high-quality results to the user, a thresholding mechanism cutting of low confidence results is needed. We chose to work with two kinds of unsupervised thresholding methods. At first, a dynamic thresholding technique based on a simple statistical outlier analysis is used on the distribution of confidence values.

If the language model has only returned incorrect results, no answer needs to be returned since the correct answers were not present in the pre-trained model. It is not possible to apply our dynamic thresholding method here. Therefore, we additionally use a static threshold. The static threshold is learned from comparing the knowledge graph's results and the language model's results: We use the overlap between the knowledge graph results, which we know to be correct, and the language model to estimate an appropriate static confidence threshold, which then is used for all queries. For each training query, we compute the average confidence of the correct results from the knowledge graph as returned by the language model. Then again, we take the average of each query's values and take that as a static threshold.

Finally, we join the result lists of the incomplete knowledge graph with the result list of our language model-based pipeline and eliminate duplicates.

5.4 Evaluation

In this section, we describe the evaluation of KnowlyBERT on the large real-world knowledge graph Wikidata and the language model BERT based on our previously published paper at ISWC 2020 [56]. Most text in this section is cited verbatim from the original paper.

We evaluate precision and recall for 41 different relations similar to other language model-based systems [14, 89, 90] and compare against a state-of-the-art relation extraction technique using distant supervision [101] and a technique for knowledge graph completion which uses high dimensional embeddings [82]. In detail, we provide an overview of the performance on different relations and provide an extensive discussion on the drawbacks and advantages of language model-based techniques for

on-the-fly query answering compared to existing techniques, which are exceptionally trained for inducing new triples in incomplete knowledge graphs.

5.4.1 Experimental Setup

Baselines. KnowlyBERT performs query answering on incomplete knowledge graphs, which may be seen as an on-the-fly knowledge graph completion method. Since no directly comparable baselines are available, we compare them to standard knowledge graph completion techniques that work in an offline setting. Here, inferring new triples using external knowledge by relation extraction from text and triple induction by structural methods purely on the knowledge graph are the most popular methods.

Therefore as a first baseline, we use a recent distant supervised relation extraction system from [101] with available pre-trained models for Wikipedia triple extraction. This baseline has already been used by Petroni et al. [90] to compare to their language model-based approach. We have used their pre-trained Wikipedia model for extracting triples from natural language text and performed relation extraction from T-Rex [29]. T-Rex links Wikidata entities and triples to Wikipedia abstracts. These linked entities in the text are used as an input for the relation extraction framework to extract triples from sentences.

As a second baseline, we have compared to another state-of-the-art technique for coping with incomplete knowledge graphs [81]. Knowledge graph embeddings are latent machine learning models for knowledge graph completion. High dimensional vector representations of entities and relations are learned from an existing knowledge graph as described in Section 4.2. Hence, it is also possible to find most likely substitutions for subject-predicate-pairs or predicate-object pairs. In our case, we use HoLE as a baseline, which has shown good results in benchmark datasets and also is scalable to the size of our large Wikidata dataset [82]. Due to the size of Wikidata, we trained HoLE using 50 dimensions for 200 epochs. Since HoLE itself only provides a top-k list of newly inferred triples ordered by their prediction probability, we only took the predictions with the best possible prediction value of HoLE. It may also include several predictions showing the same prediction value.

Dataset. Our experiments are performed on the Wikidata Truthy dump from February 6th, 2020. We evaluate only on triples where subject and object are entities having an `rdf:label` relation. For simplicity reasons, we also restrict to labels consisting of at most three words. We restrict to the 41 different relations that are used in the LAMA probe [90]. However, we use different queries since they were restricted to entities consisting of single word labels only.

We have sampled queries for each of these 41 relations by randomly choosing triples from the Wikidata. We remove the subject, creating an entity-centric SPARQL query, asking for a subject entity (?x, p, o), or removing the object respectively to ask for the object (s, p, ?x). Hence, we created 100 subject and 100 object queries for each relation, if possible. For some relations, we could only generate fewer queries. Overall, this leads to 6649 queries.

For all queries, we assume that the current Wikidata version is the ideal knowledge graph. The incomplete knowledge graph is simulated by leaving out existing triples

Table 5.1: Precision (Prec) and Recall (Rec) from KnowlyBERT against two baseline systems in percent. Relation extraction (RE) and the knowledge graph embedding technique HoLE (KE) on 41 relations. We evaluate different query parameters.

Evaluation	Parameter	Statistics		RE		KE		KnowlyBERT	
		#Queries	#Rel	Prec	Rec	Prec	Rec	Prec	Rec
Cardinality	1-1	400	2	5.5	5.5	<0.1	**20.2**	**16.9**	3.0
	1-n	3756	23	18.8	**17.4**	<0.1	11.5	**55.0**	13.7
	n-m	2493	16	16.4	19.8	<0.1	**22.6**	**36.0**	5.9
Query Type	$(s, p, ?x)$	4029	41	37.5	17.3	<0.1	**20.5**	**51.0**	16.5
	$(?x, p, o)$	2620	41	6.9	**17.9**	<0.1	9.5	**10.5**	0.3
Words	single	2474	41	39.6	13.9	<0.1	21.1	**59.6**	**25.9**
	multi	4175	41	**13.0**	**19.7**	<0.1	13.2	11.4	0.8
#Results	1	3497	41	40.5	13.2	<0.1	15.8	**51.3**	**17.4**
	$2-10$	1367	39	18.7	**20.5**	<0.1	20.4	**37.0**	4.9
	$11-100$	796	37	7.4	**30.7**	0.2	24.7	**15.8**	0.1
	>100	989	37	**5.7**	**18.2**	<0.1	4.8	<0.1	<0.1
Total		6649	41	17.5	**17.6**	<0.1	16.2	**47.5**	10.1

by performing a *leave k out* evaluation, deleting at least one and at most 100 answers from Wikidata's answer set of each query. To be comparable to the relation extraction baseline, which extracts triples from the text, we have restricted the deleted triples to triples that occur in the text corpus we use. It gives an advantage to the baseline system since it ensures that it is possible to achieve 100% recall, which is not necessarily valid for our system. The ideal knowledge graph has 54,056,746 triples, and the incomplete knowledge graph has 125,213 fewer triples deleted for the 6,649 queries. Thus, the incomplete knowledge graph comprises 53,931,533 triples.

Evaluation Metrics. We have evaluated every query separately by querying the language model and removing the answer triples that already were in the incomplete knowledge graph. For the remaining additional results, we computed precision and recall values. The reported results are average precision and recall values over all queries that returned additional results.

Implementation Details. Our system KnowlyBERT is implemented in Python 3 and is openly available on Github[2]. We also share scripts for reproducing these results available. Our system is based on the masked language model BERT from Google [25]. We use the large and cased model pre-trained by Google comprising 340m parameters. Since our system is built on the LAMA framework by Petroni et al., we are able to include arbitrary language models[3]. For the relation extraction baseline, we use the original implementation on Github[4]. The knowledge graph embedding HoLE is implemented in OpenKE [42].

[2] `https://github.com/JanKalo/KnowlyBERT`
[3] `https://github.com/facebookresearch/LAMA`
[4] `https://github.com/UKPLab/emnlp2017-relation-extraction`

5.4.2 Experimental Results

An overview of precision and recall of KnowlyBERT and the two baseline systems is presented in Table 5.1. First, we have a look at the total precision and recall values depicted in the last row. KnowlyBERT outperforms the two other approaches by more than 30% by achieving an average precision of 47.5%. In contrast to the relation extraction baseline (RE), we improve the precision drastically. However, the recall of our approach is slightly lower with 10.1% in comparison to 17.6% of the RE baseline.

HoLE (KE) shows promising results regarding the recall, but its precision is extremely low at around 0.03%. This low precision but high recall value is due to many false positives, all having top prediction values. The result for a knowledge graph embedding technique confirms recent research results that it is not ready for completing tasks in real-world knowledge graphs [2]. We present the results here anyways for completeness reasons but do not discuss them in detail. Our primary focus in this evaluation compares the RE baseline with KnowlyBERT.

In the first rows, we present the results ordered by the cardinality of the relations in the query. We have analyzed two 1-1 relations[5], 23 1-n relations and 16 n-m relations. KnowlyBERT shows its best results for 1-n relations with a precision of 55.0% and recall of 13.7%. Similarly, the two baselines show their best precision here.

We also present an evaluation of subject vs. object-based queries. KnowlyBERT achieves an extremely high precision for (s, p, ?x) queries asking for the object, but low precision and recall for queries asking for the subject of a triple. Also, the RE baseline shows a much smaller precision here but at least shows good recall values.

The next part of our evaluation presents how well the different approaches deal with multi-word entities. We analyze whether the respective results of queries that only return single word entities against queries which correct answers also comprise multi-word entities. Here, we observe that KnowlyBERT works best for single-word entities, as does the RE baseline. Multi-word entities are often much harder to find using a language model-based approach. One reason is that queries asking for persons are often multi-word queries. Answering such person queries is extremely difficult since the set of possible correct answers is often huge.

The following evaluation clusters queries by the number of results they have in the ideal knowledge graph. Here, we see that queries with few results generally show much better results. The precision and recall of KnowlyBERT for queries with a single result are over 50%, with a recall of over 17% achieving the best results. Queries with large result sets are only answered with a low result quality. If they have more than 100 answers, we hardly find any correct answer, resulting in poor precision. The RE baseline also has worse results for queries with many results but at least returns some results.

In Table 5.2, we present the results for some excellent working and poorly working relations. We see that for some relations, we achieve a precision of over 90% and a recall also above 70%. We see that many of the well working relations are about locations or languages. On the other hand, we also have several relations with an

[5]We follow the categorization of Petroni et al.[90]. Note that some queries for 1-1 relations have more than a single result.

Table 5.2: Precision (Prec) and Recall (Rec) of KnowlyBERT and the baseline systems for a variety of relations from Wikidata in percent.

Relation	Label	Statistics	RE		KE		KnowlyBERT	
		#Queries	Prec	Rec	Prec	Rec	Prec	Rec
P17	`country`	145	16.6	16.7	<0.1	21.6	**97.4**	**51.0**
P19	`birthplace`	191	21.8	**19.4**	<0.1	13.7	**73.3**	11.5
P31	`instance of`	152	**11.9**	15.0	<0.1	**17.3**	<0.1	<0.1
P36	`capital`	200	5.5	11.1	<0.1	**23.0**	**15.4**	3.0
P101	`field of work`	174	11.0	9.3	<0.1	**12.1**	**45.1**	7.8
P103	`native language`	117	<0.1	<0.1	<0.1	31.5	**100**	**74.3**
P108	`employer`	173	17.1	3.2	<0.1	**17.3**	**100**	0.6
P159	`headquarter`	190	19.6	**26.8**	<0.1	9.5	**56.8**	13.2
P279	`subclass of`	197	6.8	**28.8**	<0.1	13.5	16.7	<0.1
P1303	`instrument`	128	**35.0**	**43.4**	<0.1	15.8	<0.1	<0.1
P1412	`language spoken`	124	6.4	2.5	<0.1	**21.9**	**45.8**	17.7

extremely low recall near to 0%. Particularly poor results are shown by `instance of` and `subclass of` relations. It implies that the type of information is hardly represented in the language model. But also the `instrument` relation shows extremely poor results. In contrast, the RE baseline shows its best results here.

The evaluation of KnowlyBERT compared to other techniques for coping with incomplete knowledge graphs has shown us that none of the existing techniques is ready to deal with all problems that come with missing information. While the knowledge graph embedding-based technique has shown poor results in the real-world scenario as already shown in recent research on the evaluation of such techniques [2], the state-of-the-art relation extraction technique has shown a consistently moderate result quality with a precision and recall of around 17%.

In contrast, a language model-based approach shows a much higher precision with a slight loss in recall. We have seen that the language model has different quality depending on the relation used in the queries. In some cases, we achieve almost perfect results with over 90% precision and high recall values, whereas, for other relations, we cannot find any correct results at all. Particularly geographic relations show good results, outperforming the baselines by far. Queries with single-word entities are also showing good quality. However, multi-word entities are complicated to predict. Multi-word queries strongly correlate to queries with large result sets and subject-queries. One possible problem is that subject queries and multi-word queries often ask for long-tail entities. For these, the language model is rarely able to provide correct answers. All of these problems are reflected by our lower recall in contrast to the baselines. Particularly the relation extraction baseline still achieves an acceptable recall for these complex query types.

Note that queries with large result sets are substantially more difficult to solve. Because we do not count predicted result entities that already are in the incomplete KG, we add another difficulty. Even though a technique finds correct results, its precision for such queries might be 0%.

5.5 Conclusion

In this chapter, we presented our system KnowlyBERT, a hybrid querying answering system combining knowledge graphs with neural language models. We have shown that the idea of using a language model as an implicit store of knowledge indeed improves the querying capabilities on incomplete and heterogeneous knowledge graphs.

We have compared KnowlyBERT to other systems that could work as implicit knowledge storage: A knowledge graph embedding also stores knowledge implicitly in high dimensional latent vector spaces and can be used similarly for simple entity queries. On the other hand, relation extraction directly works on text to extract knowledge on-the-fly. We have seen that both baselines, however, show poor quality for answering queries. The knowledge graph embedding technique has massive issues in answering any query at all. The relation extraction baseline only shows a precision of under 20%. In contrast, the combination of explicit knowledge stored in a knowledge graph and implicit knowledge stored in a language model leads to a high precision of almost 50%.

Of course, these results are only preliminary when it comes to using language models instead of explicit knowledge graphs. The quality of our query answers is far from being of large help in practical applications. However, we believe that it is a first step in an important direction that can be used to overcome classical heterogeneity issues.

For now, KnowlyBERT is restricted to simple entity-centric queries. We plan to extend KnowlyBERT for more complex query types, such as join queries with multiple BGP, union queries, and aggregation queries. Here, existing datasets for question answering could serve as training data for learning templates for more complex query types. Another idea would be to fine-tune the language model to answer more complex query types similar to the work in [108]. The querying answering performance of the language models is significantly different from one relation to another. For the future, it might be interesting to investigate the performance of different queries in more detail. A detailed analysis of which knowledge was already covered in the training data text corpus would be very interesting. Additionally, the suitability of KnowlyBERT in more specific domains should be further investigated. It could be performed by evaluating different datasets than Wikidata using domain-specific language models, such as BioBert and SciBert.

6

Conclusion and Future Work

Semantic Web technologies have led to an impressive increase of entity knowledge being available online. Today, under the term *knowledge graph*, billions of facts are available to support a plethora of modern AI applications [85]. They support search, question answering, product search, and social networks. Advances in machine learning further push the growth of today's knowledge graphs due to automatic and semi-automatic information extraction techniques from text, tables, and other data sources. However, the long-known problem of *representation heterogeneity* is getting more and more problematic the larger these knowledge repositories become.

In this work, we have surveyed methods for solving heterogeneity issues throughout the previous decades. We have given an overview of schema matching, entity matching, and ontology matching. Furthermore, we provided a classification of different heterogeneity issues in real-world knowledge graphs. We gave a broad overview of state-of-the-art methods for solving these issues.

We came to the conclusion that several existing matching techniques are only evaluated in very artificial matching scenarios to resolve representation heterogeneity in real-world knowledge graphs. Particular problems that have been hardly tackled were single-knowledge graph heterogeneities and multi-knowledge graph heterogeneities with more than two knowledge graphs.

Multi-Knowledge Graph Heterogeneity. We have seen how multi-knowledge graph scenarios cause quality problems for standard instance matching systems. Standard instance matching systems have not been ready for reliably resolving entity heterogeneity issues in real-word matching scenarios. We believe that our work is an important step in recognizing that multi-knowledge graph scenarios are significantly more difficult than matching scenarios with two knowledge graphs. We have presented four techniques to improve the quality of arbitrary matching systems by more than 10% precision. It is an important step to improve the quality of overall instance matching systems.

However, up till now, many instance matching systems that are published at top conferences are only evaluated on very artificial two knowledge graph matching

scenarios [106]. Integrating our work into these systems would push research closer to real-world matching scenarios. Lately, also other researchers have published work in the direction of our work, looking at the transitive closure of matching systems [94]. Those very recent works show that the quality problems in multi-knowledge graph matching scenarios are far from being solved and still offer a wide range of possible future works.

The idea of multi-knowledge graph heterogeneities is not restricted to instance matching. For future work, our ideas should be carried over to relation and class matching techniques as well to improve their matching quality in real-world matching scenarios.

Single-Knowledge Graph Heterogeneity. We have discussed that heterogeneity issues in single-knowledge graph environments are a very scarce area of research. Only some techniques for resolving relation heterogeneity issues exist in this field. As discussed in Chapter 2, existing techniques for detecting synonymous relations are not achieving high-quality results or often make several assumptions about how the knowledge graph looks like. To the best of our knowledge, our work proposes the first methods for identifying synonymous relations in knowledge graphs purely data-driven, without making any assumptions on the underlying data. Thus, our techniques are perfectly suited to work on arbitrary, real-world knowledge graphs. We have shown that synonymous relation detection is possible with high precision, outperforming a current baseline system.

The big problem of relation heterogeneity in knowledge graphs is far from being solved automatically. Also, the techniques proposed by us still have several difficulties in identifying all synonymous relations in a purely data-driven way. With new heterogeneity types that come with huge multilingual real-world knowledge graphs, these quality problems are becoming even more severe. Thus, we believe that today, the best results could be achieved by combining human intelligence with automatic filtering techniques similar to what we presented in Section 4.3.

For the future, we believe that matching problems can be significantly improved by combining human intelligence with interpretable automatic methods, e.g., a rule mining-based heterogeneity detection approach. Furthermore, heterogeneity in single-knowledge graphs is not restricted to synonymous relations. It would be interesting to further investigate single-knowledge graph heterogeneity issues, such as duplicate entities and classes, hypernyms, hyponyms, and inverse relations. Our proposed techniques could be further extended to these scenarios as a first step.

Language Models as Knowledge Graphs. In contrast to the two previous problems, we think that the complete field of representation heterogeneity can largely profit from novel deep learning techniques. Particularly large neural language models have shown promising results in knowledge-intensive applications. In Chapter 5, we have seen how to combine a language model and a knowledge graph to overcome incomplete results due to heterogeneity in knowledge graphs. Our system is one of the early works that show that this combination can lead to high-quality querying results.

Since this work is only a first step into the usage of language models to overcome heterogeneity issues in knowledge graphs, there are a plethora of ideas that could be performed for the future. First of all, language models can be included in current matching systems and improve existing similarity metrics by adding more semantic knowledge. Also, integrating a language model into a query expansion process to overcome several heterogeneity issues could be very interesting. We believe that one of the most promising ideas is a further development of a hybrid system of a knowledge graph and a language model.

In general, this thesis has shown that evaluation scenarios for matching systems often do not reflect the problems we have in real-world knowledge graphs. Our work can be seen as a first step in the direction of solving these underrepresented heterogeneity issues.

References

[1] Ziawasch Abedjan and Felix Naumann. “Synonym Analysis for Predicate Expansion”. In: *Proceedings of the Extended Semantic Web Conference (ESWC)*. 2013, pp. 140–154.

[2] Farahnaz Akrami, Mohammed Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. “Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study”. In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2020, pp. 1995–2010.

[3] Ayman Alserafi, Alberto Abelló, Oscar Romero, and Toon Calders. “Keeping the Data Lake in Form: Proximity Mining for Pre-Filtering Schema Matching”. In: *ACM Transactions on Information Systems (TOIS)* 38.3 (May 2020).

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. “DBpedia: A Nucleus for a Web of Open Data”. In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2007, pp. 722–735.

[5] Wouter Beek, Joe Raad, Jan Wielemaker, and Frank van Harmelen. “sameAs.cc: The Closure of 500M owl:sameAs Statements”. In: *Proceedings of the Extended Semantic Web Conference (ESWC)*. 2018, pp. 65–80.

[6] Jacob Berlin and Amihai Motro. “Database Schema Matching Using Machine Learning with Feature Selection”. In: *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*. 2002, pp. 452–466.

[7] Tim Berners-Lee. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco, 1999.

[8] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (2001), pp. 34–43.

[9] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. “Generic schema matching, ten years later”. In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2011, pp. 695–701.

[10] Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. “LINDA: Distributed Web-of-Data-Scale Entity Matching”. In: *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. 2012, pp. 2104–2108.

[11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge". In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2008, pp. 1247–1250.

[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating Embeddings for Modeling Multi-relational Data". In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. 2013, pp. 2787–2795.

[13] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella. "An Entity Name System (ENS) for the Semantic Web". In: *Proceedings of the Extended Semantic Web Conference (ESWC)*. 2008, pp. 258–272.

[14] Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. "Inducing Relational Knowledge from BERT". In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2020, pp. 7456–7463.

[15] Dan Brickley and R.V. Guha. *RDF Schema 1.1. W3C Recommendation*. Feb. 2014.

[16] T. Brown et al. "Language Models are Few-Shot Learners". In: *ArXiv* abs/2005.14165 (2020).

[17] Fei Chang, Guowei Chen, and Songmao Zhang. "FCAMap-KG Results for OAEI 2019". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2019, pp. 138–145.

[18] Michelle Cheatham and Pascal Hitzler. "The properties of property alignment." In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2014, pp. 13–24.

[19] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. "Reading Wikipedia to Answer Open-Domain Questions". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL))*. 2017, pp. 1870–1879.

[20] Guowei Chen and Songmao Zhang. "Identifying Mappings among Knowledge Graphs by Formal Concept Analysis". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. Vol. 2536. 2019, pp. 25–35.

[21] Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Ian Horrocks. "Canonicalizing Knowledge Base Literals". In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2019, pp. 110–127.

[22] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 1511–1517.

[23] Weize Chen, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. "Quantifying Similarity between Relations with Fact Distribution". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL))*. 2019, pp. 2882–2894.

[24] Richard Cyganiak, David Wood, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation*. Feb. 2014.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. June 2019, pp. 4171–4186.

[26] Li Ding, Joshua Shinavier, Zhenning Shangguan, and Deborah L. McGuinness. "SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in Linked Data". In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2010, pp. 145–160.

[27] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. "Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 2014, pp. 601–610.

[28] Lisa Ehrlinger and Wolfram Wöß. "Towards a Definition of Knowledge Graphs." In: *SEMANTiCS* (2016), pp. 1–4.

[29] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. "T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[30] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching, 2nd Edition*. 2013.

[31] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. "The AgreementMakerLight Ontology Matching System". In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. 2013, pp. 527–541.

[32] Daniel Faria, Catia Pesquita, Teemu Tervo, Francisco M. Couto, and Isabel F. Cruz. "AML and AMLC Results for OAEI 2019". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. Vol. 2536. 2019, pp. 101–106.

[33] Alflo Ferrara, Davide Lorusso, Stefano Montanelli, and Gaia Varese. "Towards a Benchmark for Instance Matching". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2008, pp. 37–48.

[34] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M. Suchanek. "Canonicalizing Open Knowledge Bases". In: *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. 2014, pp. 1679–1688.

[35] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. "Fast rule mining in ontological knowledge bases with AMIE+". In: *The Very Large Database (VLDB) Journal* 24.6 (Dec. 2015), pp. 707–730.

[36] Luis A. Galárraga, Nicoleta Preda, and Fabian M. Suchanek. "Mining rules to align knowledge bases". In: *Proceedings of the Workshop on Automated Knowledge Base Construction (AKBC)*. 2013, pp. 43–48.

[37] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. "AMIE: Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases". In: *Proceedings of the International Conference on the World Wide Web (TheWebConf)*. 2013, pp. 413–422.

[38] Michelle Girvan and Mark EJ Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826.

[39] Binbin Gu, Zhixu Li, Xiangliang Zhang, An Liu, Guanfeng Liu, Kai Zheng, and Lei Zhao. "The Interaction Between Schema Matching and Record Matching in Data Integration". In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 29.1 (2017), pp. 186–199.

[40] Alon Halevy. "Why your data won't mix". In: *ACM Queue* 3.8 (2005), pp. 50–58.

[41] Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. "When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data". In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2010, pp. 305–320.

[42] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. "OpenKE: An Open Toolkit for Knowledge Embedding". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, pp. 139–144.

[43] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J. Miller. "Framework for Evaluating Clustering Algorithms in Duplicate Detection". In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2009, pp. 1282–1293.

[44] Oktie Hassanzadeh and Mariano P Consens. "Linked Movie Data Base". In: *Workshop on Linked Data on the Web (LDOW)*. 2009.

[45] Sven Hertling and Heiko Paulheim. "DOME results for OAEI 2018". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2018, pp. 144–151.

[46] Sven Hertling and Heiko Paulheim. "DOME Results for OAEI 2019". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2019, pp. 123–130.

[47] Sven Hertling and Heiko Paulheim. "The Knowledge Graph Track at OAEI". In: *Proceedings of the Extended Semantic Web Conference (ESWC)*. 2020, pp. 343–359.

[48] Silviu Homoceanu, Jan-Christoph Kalo, and Wolf-Tilo Balke. "Putting Instance Matching to the Test: Is Instance Matching Ready for Reliable Data Linking?" In: *Foundations of Intelligent Systems (ISMIS)*. 2014, pp. 274–284.

[49] Prateek Jain, Pascal Hitzler, Amit P. Sheth, Kunal Verma, and Peter Z. Yeh. “Ontology Alignment for Linked Open Data”. In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2010, pp. 402–417.

[50] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. “Knowledge Graph Embedding via Dynamic Mapping Matrix”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. 2015, pp. 687–696.

[51] Zhengbao Jiang, Jun Araki, Donghan Yu, Ruohong Zhang, Wei Xu, Yiming Yang, and Graham Neubig. “Learning Relation Entailment with Structured and Textual Information”. In: *Automated Knowledge Base Construction (AKBC)*. 2020.

[52] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. “How Can We Know What Language Models Know?” In: *Transactions of the Association for Computational Linguistics (TACL)* 8 (2020), pp. 423–438.

[53] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. “LogMap: Logic-Based and Scalable Ontology Matching”. In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2011, pp. 273–288.

[54] Jan-Christoph Kalo. “Analysing the Problem of Transitivity in Instance Matching Tasks on Linked Data”. `http://www.ifis.cs.tu-bs.de/node/3262`. MA thesis. Carl-Friedrich-Gauß Fakultät, Technische Universität Braunschweig, 2014.

[55] Jan-Christoph Kalo, Philipp Ehler, and Wolf-Tilo Balke. “Knowledge Graph Consolidation by Unifying Synonymous Relationships”. In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2019, pp. 276–292.

[56] Jan-Christoph Kalo, Leandra Fichtel, Philipp Ehler, and Wolf-Tilo Balke. “KnowlyBERT - Hybrid Query Answering over Language Models and Knowledge Graphs”. In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2020, pp. 294–310.

[57] Jan-Christoph Kalo, Silviu Homoceanu, Jewgeni Rose, and Wolf-Tilo Balke. “Avoiding Chinese Whispers: Controlling End-to-End Join Quality in Linked Open Data Stores”. In: *Proceedings of the ACM Web Science Conference (WebSci)*. 2015, pp. 1–10.

[58] Jan-Christoph Kalo, Stephan Mennicke, Philipp Ehler, and Wolf-Tilo Balke. “Detecting Synonymous Properties by Shared Data-Driven Definitions”. In: *Proceedings of the Extended Semantic Web Conference (ESWC)*. 2020, pp. 360–375.

[59] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. “Dense Passage Retrieval for Open-Domain Question Answering”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 6769–6781.

[60] William Kent. “Many forms of a single fact.” In: *Digest of Papers - IEEE Computer Society International Conference* (1989), pp. 438–443.

[61] Won Kim and Jungyun Seo. “Classifying Schematic and Data Heterogeneity in Multidatabase Systems”. In: *Computer* 24.12 (Dec. 1991), pp. 12–18.

[62] loannis Koumarelas, Thorsten Papenbrock, and Felix Naumann. “MDedup: Duplicate Detection with Matching Dependencies”. In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2020, pp. 712–725.

[63] Amir Laadhar, Faiza Ghozzi, Imen Megdiche, Franck Ravat, Olivier Teste, and Faiez Gargouri. “POMap++ Results for OAEI 2019: Fully Automated Machine Learning Approach for Ontology Matching”. In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2019, pp. 169–174.

[64] Simon Lacoste-Julien, Konstantina Palla, Alex Davies, Gjergji Kasneci, Thore Graepel, and Zoubin Ghahramani. “SIGMa: Simple Greedy Matching for Aligning Large Knowledge Bases”. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 2013, pp. 572–580.

[65] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. “Rimom: A dynamic multistrategy ontology alignment framework”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 21.8 (2008), pp. 1218–1232.

[66] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2015, pp. 2181–2187.

[67] Hanxiao Liu, Yuexin Wu, and Yiming Yang. “Analogical Inference for Multi-relational Embeddings”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2017, pp. 2168–2178.

[68] R Duncan Luce. “Semiorders and a theory of utility discrimination”. In: *Econometrica, Journal of the Econometric Society* (1956), pp. 178–191.

[69] Xin Mao, Wenting Wang, Huimin Xu, Man Lan, and Yuanbin Wu. “Mraea: An efficient and robust entity alignment approach for cross-lingual knowledge graph”. In: *Proceedings of the International Conference on Web Search and Data Mining (WSDM)* (2020), pp. 420–428.

[70] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview, W3C Recommendation*. Feb. 2004.

[71] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. 2013, pp. 3111–3119.

[72] George A. Miller. “WordNet: A Lexical Database for English”. In: *Communications of the ACM (CACM)* 38.11 (Nov. 1995), pp. 39–41.

[73] Tova Milo and Sagit Zohar. "Using Schema Matching to Simplify Heterogeneous Data Translation". In: *Proceedings of the Very Large Databases Conference (VLDB)*. 1998, pp. 122–133.

[74] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. "Deep Learning for Entity Matching: A Design Space Exploration". In: *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2018, pp. 19–34.

[75] Felix Naumann. "Data Profiling Revisited". In: *ACM SIGMOD Record* 42.4 (Feb. 2014), pp. 40–49.

[76] Felix Naumann and Melanie Herschel. "An introduction to duplicate detection". In: *Synthesis Lectures on Data Management* 2.1 (2010), pp. 1–87.

[77] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004).

[78] Khai Nguyen and Ryutaro Ichise. "SLINT+ Results for OAEI 2013 Instance Matching". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2013, pp. 177–183.

[79] Khai Nguyen, Ryutaro Ichise, and Bac Le. "SLINT: a schema-independent linked data interlinking system". In: *Proceedings of the International Workshop on Ontology Matching at International Semantic Web Conference (OM)*. 2012, pp. 1–12.

[80] Quoc Viet Hung Nguyen, Thanh Tam Nguyen, Zoltán Miklós, Karl Aberer, Avigdor Gal, and Matthias Weidlich. "Pay-as-you-go reconciliation in schema matching networks". In: *Proceedings - International Conference on Data Engineering* (2014), pp. 220–231.

[81] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 11–33.

[82] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. "Holographic Embeddings of Knowledge Graphs". In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2016, pp. 1955–1961.

[83] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A Three-way Model for Collective Learning on Multi-relational Data". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2011, pp. 809–816.

[84] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "Factorizing YAGO". In: *Proceedings of the International Conference on the World Wide Web (TheWebConf)*. 2012, pp. 271–280.

[85] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. "Industry-Scale Knowledge Graphs: Lessons and Challenges". In: *Communications of the ACM (CACM)* 62 (July 2019), pp. 36–43.

[86] Aris M. Ouksel and Amit Sheth. "Semantic interoperability in global information systems". In: *ACM Sigmod Record* 28.1 (1999), pp. 5–12.

[87] Shichao Pei, Lu Yu, Guoxian Yu, and Xiangliang Zhang. "REA: Robust Cross-Lingual Entity Alignment Between Knowledge Graphs". In: *Proceedings of the 26th International Conference on Knowledge Discovery & Data Mining*. KDD '20. 2020, pp. 2175–2184.

[88] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[89] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. "How Context Affects Language Models' Factual Predictions". In: *Automatic Knowledge Base Construction (AKBC)*. 2020.

[90] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. "Language Models as Knowledge Bases?" In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Nov. 2019, pp. 2463–2473.

[91] Nina Poerner, Ulli Waltinger, and Hinrich Schütze. "E-BERT: Efficient-Yet-Effective Entity Embeddings for BERT". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 803–818.

[92] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF. W3C Recommendation*. Jan. 2008.

[93] Joe Raad, Wouter Beek, Frank van Harmelen, Nathalie Pernelle, and Fatiha Saïs. "Detecting Erroneous Identity Links on the Web Using Network Metrics". In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2018, pp. 391–407.

[94] Joe Raad, Wouter Beek, Frank van Harmelen, Jan Wielemaker, Nathalie Pernelle, and Fatiha Saïs. "Constructing and Cleaning Identity Graphs in the LOD Cloud". In: *Data Intelligence* 2.3 (2020), pp. 323–352.

[95] Joe Raad, Nathalie Pernelle, Fatiha Saïs, Wouter Beek, and Frank van Harmelen. "The sameAs Problem: A Survey on Identity Management in the Web of Data". In: *CoRR* abs/1907.10528 (2019).

[96] Erhard Rahm and Philip A. Bernstein. "A survey of approaches to automatic schema matching". In: *VLDB Journal* 10.4 (2001), pp. 334–350.

[97] John W Ratcliff and David E Metzener. "Pattern-matching-the gestalt approach". In: *Dr Dobbs Journal* 13.7 (1988), p. 46.

[98] Peter J. Rousseeuw and Mia Hubert. "Robust statistics for outlier detection". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 73–79.

[99] Chao Shao, Lin-Mei Hu, Juan-Zi Li, Zhi-Chun Wang, Tonglee Chung, and Jun-Bo Xia. "RiMOM-IM: A Novel Iterative Framework for Instance Matching". In: *Journal of Computer Science and Technology* 31 (Jan. 2016), pp. 185–197.

[100] Amit Singhal. *Introducing the knowledge graph: things, not strings*. Retrieved on 01.03.2021. 2012. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[101] Daniil Sorokin and Iryna Gurevych. "Context-Aware Representations for Knowledge Base Relation Extraction". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 1784–1789.

[102] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. "PARIS: Probabilistic Alignment of Relations, Instances, and Schema". In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2011, pp. 157–168.

[103] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge". In: *Proceedings off the International Conference on the World Wide Web (TheWebConf)*. 2007, pp. 697–706.

[104] Zequn Sun, Wei Hu, and Chengkai Li. "Cross-lingual entity alignment via joint attribute-preserving embedding". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2017), pp. 628–644.

[105] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. "Bootstrapping entity alignment with knowledge graph embedding". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018, pp. 4396–4402.

[106] Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. "A benchmarking study of embeddingbased entity alignment for knowledge graphs". In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2020, pp. 2326–2340.

[107] *The Semantic Web Stack*. Retrieved on 01.03.2021. URL: https://en.wikipedia.org/wiki/Semantic_Web_Stack.

[108] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. "Neural Databases". In: (2020).

[109] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. "Entity alignment between knowledge graphs using attribute embeddings". In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2019, pp. 297–304.

[110] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex Embeddings for Simple Link Prediction". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2016, pp. 2071–2080.

[111] Stijn Marinus Van Dongen. "Graph clustering by flow simulation". PhD thesis. 2000.

[112] Shikhar Vashishth, Prince Jain, and Partha Talukdar. "CESI: Canonicalizing Open Knowledge Bases Using Embeddings and Side Information". In: *Proceedings of the International Conference on the World Wide Web (TheWebConf)*. 2018, pp. 1317–1327.

[113] Denny Vrandečić and Markus Krötzsch. “Wikidata: a free collaborative knowledgebase”. In: *Communications of the ACM (CACM)* 57.10 (2014), pp. 78–85.

[114] Jiannan Wang, Guoliang Li, Jeffrey Xu Yu, and Jianhua Feng. “Entity matching: How similar is similar”. In: *Proceedings of the Very Large Databases Conference (VLDB)*. 2011, pp. 622–633.

[115] Q. Wang, Z. Mao, B. Wang, and L. Guo. “Knowledge Graph Embedding: A Survey of Approaches and Applications”. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 29.12 (2017), pp. 2724–2743.

[116] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Guihong Cao, Daxin Jiang, and Ming Zhou. *K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters*. 2020.

[117] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. “Knowledge Graph Embedding by Translating on Hyperplanes”. In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2014, pp. 1112–1119.

[118] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. “Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, pp. 349–357.

[119] Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. “Learning to Distinguish Hypernyms and Co-Hyponyms”. In: *Proceedings of the International Conference on Computational Linguistics (COLING)*. 2014, pp. 2249–2259.

[120] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. “Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[121] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

[122] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. “ERNIE: Enhanced Language Representation with Informative Entities”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL))*. 2019, pp. 1441–1451.

[123] Ziqi Zhang, Anna Lisa Gentile, Isabelle Augenstein, Eva Blomqvist, and Fabio Ciravegna. “Mining equivalent relations from linked data”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL))*. 2013, pp. 289–293.

[124] Ziqi Zhang, Anna Lisa Gentile, Eva Blomqvist, Isabelle Augenstein, and Fabio Ciravegna. “An unsupervised data-driven method to discover equivalent relations in large linked datasets”. In: *Semantic web* 8.2 (2017), pp. 197–223.

[125] Ziqi Zhang, Anna Lisa Gentile, Eva Blomqvist, Isabelle Augenstein, and Fabio Ciravegna. "Statistical Knowledge Patterns: Identifying Synonymous Relations in Large Linked Datasets". In: *Proceedings of the International Semantic Web Conference (ISWC)*. 2013, pp. 703–719.

[126] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. "Iterative Entity Alignment via Joint Knowledge Embeddings". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 4258–4264.

[127] Yan Zhuang, Guoliang Li, Zhuojian Zhong, and Jianhua Feng. "Hike: A Hybrid Human-Machine Method for Entity Alignment in Large-Scale Knowledge Bases". In: *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. 2017, pp. 1917–1926.